Universidade Federal do Rio de Janeiro

Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais

Bruno Lemos Barroso

CRIPTOGRAFIA: Aspectos Teóricos e Proposta de Autenticação Utilizando Dados Cadastrais do Usuário

Rio de Janeiro

2016

Bruno Lemos Barroso

CRIPTOGRAFIA:

Aspectos Teóricos e Proposta de Autenticação Utilizando Dados Cadastrais do Usuário

Monografia apresentada para obtenção do título de Especialista em Gerência de Redes de Computadores no Curso de Pós-Graduação Lato Sensu em Gerência de Redes de Computadores e Tecnologia Internet do Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais da Universidade Federal do Rio de Janeiro – NCE/UFRJ.

Orientador:

Claudio Miceli de Farias, D.Sc., UFRJ, Brasil

Rio de Janeiro

2016

Bruno Lemos Barroso

CRIPTOGRAFIA:

Aspectos Teóricos e Proposta de Autenticação Utilizando Dados Cadastrais do Usuário

Monografia apresentada para obtenção do título de Especialista em Gerência de Redes de Computadores no Curso de Pós-Graduação Lato Sensu em Gerência de Redes de Computadores e Tecnologia Internet do Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais da Universidade Federal do Rio de Janeiro – NCE/UFRJ.

Aprovada em março de 2016.

Claudio Miceli de Farias, D.Sc., UFRJ, Brasil

Dedico este trabalho a meus pais que procuraram sempre me dar a tranquilidade necessária para que eu pudesse priorizar meus estudos. E a minha esposa que abdicou de outras atividades para exclusivamente cuidar de nosso filho enquanto eu dedicava meu tempo às atividades deste curso.

AGRADECIMENTOS

Gostaria de agradecer aos excelentes professores do MOT que nos transmitiram seus vastos conhecimentos e a todos os colegas que chegaram ao fim desta difícil jornada sempre mantendo um clima de cordialidade, amizade e ajuda mútua.

RESUMO

BARROSO, Bruno Lemos. **CRIPTOGRAFIA: Aspectos Teóricos e Proposta de Autenticação Utilizando Dados Cadastrais do Usuário**. Monografia (Especialização em Gerência de Redes e Tecnologia Internet). Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Universidade Federal do Rio de Janeiro. Rio de Janeiro, 2016.

Estudo sobre criptografia apresentando seus principais conceitos, técnicas, algoritmos e a sua importância em um mundo digital onde informações cada vez mais valiosas trafegam pela internet estando sujeitas à interceptação por usuários maliciosos dispostos a obter vantagens (principalmente financeiras). Serão apresentadas as vantagens e desvantagens de cada técnica criptográfica e em quais situações cada uma delas é indicada.

Ao final do estudo será apresentada uma proposta de autenticação baseada no algoritmo de criptografia assimétrica RSA. A autenticação utilizará como critérios para geração de chaves criptográficas informações cadastrais do usuário (nome de pai ou mãe, cpf ou rg, data de nascimento, dentre outras) presentes em uma base de dados de uma empresa. Esta proposta tem como objetivo gerar um resultado criptográfico diferente para a senha do usuário a cada autenticação. Assim, caso esta conexão seja sendo interceptada constantemente, será simulada uma senha diferente a cada acesso (OTP – one-time password), dificultando a descoberta da mesma por potenciais atacantes.

ABSTRACT

BARROSO, Bruno Lemos. **CRIPTOGRAFIA: Aspectos Teóricos e Proposta de Autenticação Utilizando Dados Cadastrais do Usuário**. Monografia (Especialização em Gerência de Redes e Tecnologia Internet). Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Universidade Federal do Rio de Janeiro. Rio de Janeiro, 2016.

The present study about cryptography presents its main concepts, techniques, algorithms and its importance in a digital world where increasingly valuable informations travel over the internet are subject to interception by malicious users ready to get advantages (mainly financial). Advantages and disadvantages of each cryptographic technique will be presented and situations where each of them is indicated.

At eh end of study will be presented an authentication proposal based on asymmetric cryptography algorithm RSA. Authentication will use as criteria for generating cryptographic keys social user informations (such as name of father or mother, cpf or id numbers, date of birth) registered in a database of a enterprise. This proposal aims to generate a different cryptographic result for the user's password on each login. So if this connection is being intercepted constantly, it will be simulated a different password each time (OTP - One-time password), increasing the difficultof password discoveryby potential attackers.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

3DES Triple Data Encryption Standard
AES Advanced Encryption Standard
DES Data Encryption Standard
EDE Encrypt-Decrypt-Encrypt
LSB Least Significant Bit
NIST National Institute of Standards and Technology
NSA National Security Agency
OTP One-Time Password
RSA Rivest-Shamir-Adleman
SSL Secure Socket Layer

SUMÁRIO

1 **INTRODUÇÃO**

Uma das maiores características que faz o *Homo Sapiens* diferenciar-se das outras espécies é a capacidade de comunicar-se. Desde o primórdio da humanidade o homem desenvolve novas técnicas de comunicação, a começar pelas pinturas rupestres que até hoje despertam interesse para um grande número de pessoas. Com o desenvolvimento da comunicação o homem passou a transmitir, não mais da forma tradicional (comunicação verbal e presencial), informações estratégicas e valiosas. Entretanto, muitas destas informações são valiosas, não apenas para o detentor das mesmas, mas também para um grupo de pessoas que tentarão obtê-las para obter vantagens econômicas, estratégicas, militares, dentre outras.

Com o crescente número de usuários conectados à internet aumenta também a quantidade de informações sigilosas trafegando pela rede, como por exemplo dados bancários. A grande maioria destes usuários são leigos que não se preocupam ou não tem o conhecimento necessário para proteger-se de ataques virtuais. Aproveitando-se deste fato, atacantes virtuais desenvolvem ferramentas para obtenção destes dados tendo como principal objetivo o retorno financeiro. Assim, técnicas devem ser implementadas visando a proteção dos dados destes usuários de uma forma proativa, ou seja, sem a iniciativa dos mesmos.

Surgiu, então, a necessidade de criar mecanismos para que a informação, mesmo capturada, não pudesse ser decifrada por um detentor não-autorizado. Dentre as principais técnicas existentes para o mascaramento destas informações estão a esteganografia e a criptografia. A esteganografia tem como principal objetivo esconder a informação para que ela não possa ser percebida por terceiros, ou seja, a presença de mensagens escondidas dentro de arquivos (tratando-se de esteganografia digital) é simplesmente desconhecida (ARTZ,2001 apud

CHIRIGATI,KIKUCHI,GOMES,2009). Já a criptografia preocupa-se em substituir a informação original por uma diferente (codificar) de forma que apenas o destinatário desejado tenha mecanismos para acessar o conteúdo original (descodificar). As duas podem ser empregadas em conjunto como descreve (ROCHA,2003): Uma mensagem, antes de ser mascarada, é criptografada segundo uma chave. Logo após, a partir de uma chave de deslocamento, escolhe-se os bits menos significativos (*LSBs - Least Significant Bits*) que devem ser alterados na imagem de cobertura. Propõe-se a utilização da criptografia como um passo adicional antes de efetuar o mascaramento das mensagens na ferramenta desenvolvida.

Este trabalho estará concentrado na segunda técnica apresentada: a criptografia. Ela é a solução, visando a segurança digital, mais difundida atualmente. A criptografia, em relação à esteganografia, pode ser considerada uma forma de proteção que exige menos largura de banda de rede para sua comunicação porque apenas a mensagem codificada é enviada, enquanto que no outro método, além da mensagem, também é enviado um arquivo que servirá de “esconderijo” para a informação codificada (este arquivo pode ser uma foto ou vídeo). A criptografia também permite sua aplicação considerando o valor da informação a ser protegida: informações mais valiosas utilizam chaves maiores, menos valiosas chaves menores (o conceito de chave criptográfica será definido adiante). Desta maneira o custo da proteção pode ser melhor empregado de acordo com cada situação. Outro fator que contribui para a disseminação da criptografia digital é a vasta literatura e quantidade de estudos existentes sobre o tema permitindo sua constante evolução e permanente interesse sobre o tema.

Atualmente existem vários protocolos e aplicativos que vêm embutidos no sistema operacional dos computadores que têm como objetivo proporcionar

segurança aos usuários. Como são mecanismos padrão, estes eles serão aqueles que tornar-se-ão os maiores alvos para um ataque virtual porque são os que tem um maior número de usuários utilizando-os consequentemente tendo mais vítimas em potencial. Assim, é aconselhado que instituições que prestam serviços, principalmente financeiros, através de sites ou aplicativos próprios, criem seus próprios mecanismos de segurança visando a proteger as informações de seus clientes. Estes mecanismos devem partir do pressuposto que o usuário é o mais leigo possível e que não há nenhuma outra forma de proteção no computador do cliente (pior cenário).

Esta proposta apresentará uma solução factível de implementação e que aplicada junto a outros mecanismos de proteção (como protocolos de segurança e antivírus) proporcionará aos usuários um ambiente mais seguro para o acesso a suas informações bancárias. A solução terá portabilidade, pois é baseada na linguagem de programação Java, podendo ser aplicada na maioria dos sistemas operacionais atuais. Também levará em consideração a capacidade computacional dos *hardwares* que a executarão, podendo ser aplicada tanto em *notebook* e computadores *desktops* tradicionais quanto a *smartphones* (que possuem capacidade de processamento menor). Sob a lógica de programação será flexível para adaptações à realidade de qualquer instituição que deseje implementar a ideia.

O segundo capítulo apresentará os principais conceitos sobre criptografia e uma introdução sobre os métodos criptográficos mais utilizados em um ambiente computacional: criptografia simétrica e assimétrica. Nos próximos capítulos 3 e 4 serão apresentados as características, vantagens e eventuais fragilidades de cada método. No quinto capítulo será apresentada uma proposta de criptografia da senha do usuário no momento da autenticação junto à instituição financeira. A criptografia

estará baseada na escolha de informações pessoais aleatórias do usuário, ou seja, a cada autenticação será escolhida uma informação diferente, gerando, consequentemente, uma chave distinta. Assim, o resultado da senha codificada também será diferente a cada conexão. Caso um hacker venha a capturar a senha codificada em momentos diferentes será simulado um ambiente onde a senha é alterada constantemente (*One-Time Password – OTP).*

2 **CRIPTOGRAFIA: CONCEITOS E MÉTODOS**

Este capítulo apresentará uma explanação geral sobre a criptografia: sua evolução, conceitos e técnicas. Primeiramente serão apresentados os conceitos sobre o que é criptografia, um resumo sobre o seu processo de funcionamento e um exemplo de sua utilização. Na próxima seção serão explicitados alguns motivos para sua utilização e um exemplo que ilustra estes motivos. Finalizando o capítulo serão detalhados os tipos de técnicas criptográficas existentes, e os métodos desenvolvidos dentro de cada técnica. Serão apresentados métodos que abrangem desde o período medieval até os mais modernos encontrados atualmente.

2.1 DEFINIÇÕES

Criptografia vem do grego *Kriptosgraph* (*Kriptos* = escondido e *graph* = escrita). Ela tem como objetivo codificar uma informação de tal maneira que apenas o remetente e um destinatário autorizado tenham mecanismos para decodificá-la para que ela volte ao seu conteúdo original.

A ciência criptográfica depende de dois processos: a encriptação ou codificação e a decriptação ou decodificação. Para que os dois processos trabalhem em sinergia é preciso a criação de um código secreto que é partilhado tanto pelo remetente quanto pelo destinatário da mensagem. Desta maneira um interceptador da mensagem não conseguirá decifrá-la, garantindo a confidencialidade da informação. Segue ilustração deste processo onde Bruno escreve uma mensagem codificada para Fabiana. Apenas os dois possuem o código que codificou a mensagem. Eduardo interceptará a mensagem mas não conseguirá lê-la.

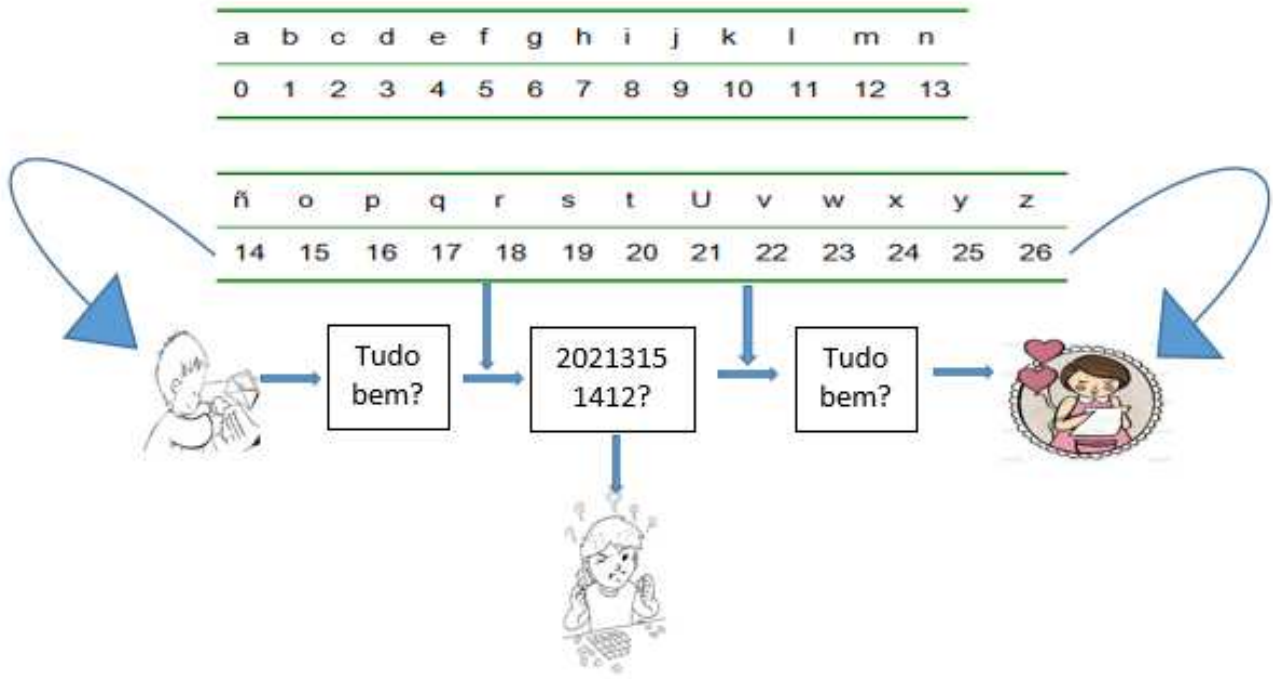


Figura 1 – Codificação e Decodificação da Mensagem

Deve ser notado que a segurança do processo acima depende da complexidade do código secreto compartilhado entre as partes. Quanto mais elaborado for o código menor será a probabilidade que a mensagem seja decodificada caso seja interceptada por terceiros durante o processo de envio.

Não existe um código que garanta 100% de certeza de que a mensagem não será decifrada (caso seja interceptada por terceiros). Os códigos podem ser avaliados quanto a sua efetividade calculando o tempo estimado que uma mensagem codificada por ele levaria para ser descoberta. Para que um código seja considerado eficaz deve ser respondida a seguinte pergunta: o custo para a quebra deste código é maior do que o valor da informação que ele está protegendo? Caso positivo ele pode ser considerado eficaz, caso negativo ele não é indicado para este tipo de informação.

Outra medida importante de segurança neste processo é a maneira como o código secreto é compartilhado. Não existe utilidade em um código complexo e difícil

de ser quebrado se a sua estratégia de transmissãofor falha. Deve-se garantir que apenas as partes interessadas tenham conhecimento do código, caso contrário uma mensagem cifrada com um código de conhecimento público terá a mesma fragilidade que uma mensagem em texto livre. Na figura abaixo, ao contrário da anterior, o código é de conhecimento também de Eduardo. Assim, ao interceptar a mensagem, o mesmo conseguirá decifrá-la.

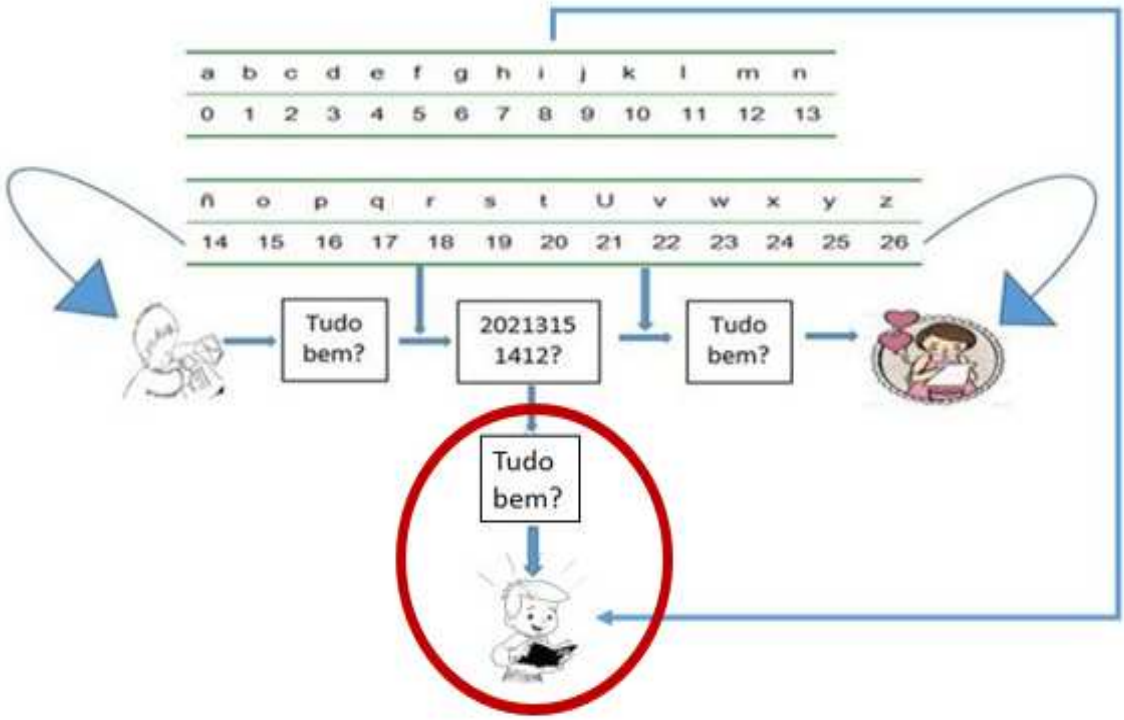


Figura 2 – Exemplo de Código com Compartilhamento Público

2.2 NECESSIDADE DE CONFIDENCIALIDADE – UMA BREVE HISTÓRIA DA CRIPTOGRAFIA

O desenvolvimento da comunicação, apesar da sua importância histórica,trouxe um dilema: como transmitir uma informação confidencial para algum local geograficamente distante sem expor-se ao risco de ela cair em domínio de adversários? Assim, em paralelo à comunicação também foi desenvolvida a codificação das mensagens.

“O primeiro uso documentado da criptografia foi em torno de 1900 a.c., no Egito, quando um escriba usou hieróglifos fora do padrão numa inscrição” (Vianna,2011).

A criptografia desenvolveu-se principalmente no início do século XX devido a dois fatores intrinsicamente ligados entre si:

I - A evolução dos meios de comunicação (serviço postal, rádio, telex, dentre outros) que permitia um maior volume de informações trafegando a grandes distâncias;

II - As duas grandes guerras mundiais.Exércitos beneficiaram-se desta evolução para transmitir ordens de batalhas para seus soldados (principalmente via rádio) mas também se viam fragilizados com a possibilidade destas ordens serem interceptadas pelos inimigos e serem usadas contra eles mesmos. Devido à necessidade de confidencialidade dos planos de guerra sendo transmitidos, novas estratégias de transmissões eram de vital importância para vencer as batalhas. Dentro deste contexto foram desenvolvidos mecanismos capazes de transmitir mensagens codificadas, criando uma guerra paralela longe dos campos de batalhas, onde de um lado técnicos criavam métodos de codificação e de outro eram criados meios de decodificar estas mensagens. O maior exemplo que ilustra esta guerra é a máquina Enigma.

Durante a primeira guerra mundial as mensagens eram codificadas ainda se utilizando de lápis e papel. Então o engenheiro alemão Arthur Scherbius em 1918 resolveu criar um método que aproveitasse a tecnologia do século XX. Daí surgiu a ideia de uma máquina de codificação utilizando rotores: a *Máquina Enigma*. Embora não tenha feito sucesso comercialmente, a máquina (com algumas alterações) foi adotada pela marinha alemã em 1926 (KHAN, 91 apud KRISCHER, 2013) e

posteriormente em 1928 (com mais alterações) pelo exército do mesmo país e finalmente em 1933 pelo serviço diplomático. O processo de codificação e decodificação das mensagens, resumidamente, era o seguinte:

> I – Diariamente eram definidos a ordem dos rotores em instruções por escrito;
>
> II - A mensagem era datilografada em um texto sem espaços diretamente na Enigma;
>
> III – A mensagem codificada era exibida em um painel luminoso;
>
> IV – A mensagem era transmitida via rádio (mesmo interceptada era inteligível para os inimigos);
>
> V – O destinatário ajustava seus rotores da mesma maneira que o remetente;
>
> VI – O destinatário datilografa a mensagem cifrada para recuperar o texto sem espaços...
>
> VII - ... que acende no painel luminoso. (FONG,2015)

A criação deste método pioneiro de codificação criou condições que levaram os alemães a expandirem seus domínios territoriais pela Europa.

Figura 3 – A máquina Enigma

Fonte: Portal UOL[1]

Mesmo com a derrota da Alemanha na primeira guerra mundial os poloneses preocupavam-se com a reorganização alemã e possíveis novos planos de ataques. Por isto, durante a década de 30, mesmo sem nenhuma guerra em andamento, matemáticos da Polônia trabalhavam e obtiveram êxito na decodificação de mensagens militares alemãs sendo considerado as primeiras pessoas a “quebrarem” a Enigma. Mas em 1938 os alemães acrescentaram inovações na máquina dificultando a tarefa dos polonesesde continuarem decifrando as mensagens. Já sem recursos e com a Polônia sendo invadida pelo exército alemão, os matemáticos poloneses viram-se obrigados a fugir do país mas antes transferiram todo seu conhecimento sobre a enigma para os aliados franceses e britânicos. (FONG,2015)

Assim, continuando os estudos dos poloneses, Alan Turing (matemático britânico) e sua equipe decifraram o código da nova Enigma, sendo fundamental para a derrocada alemã na segunda grande guerra.

---

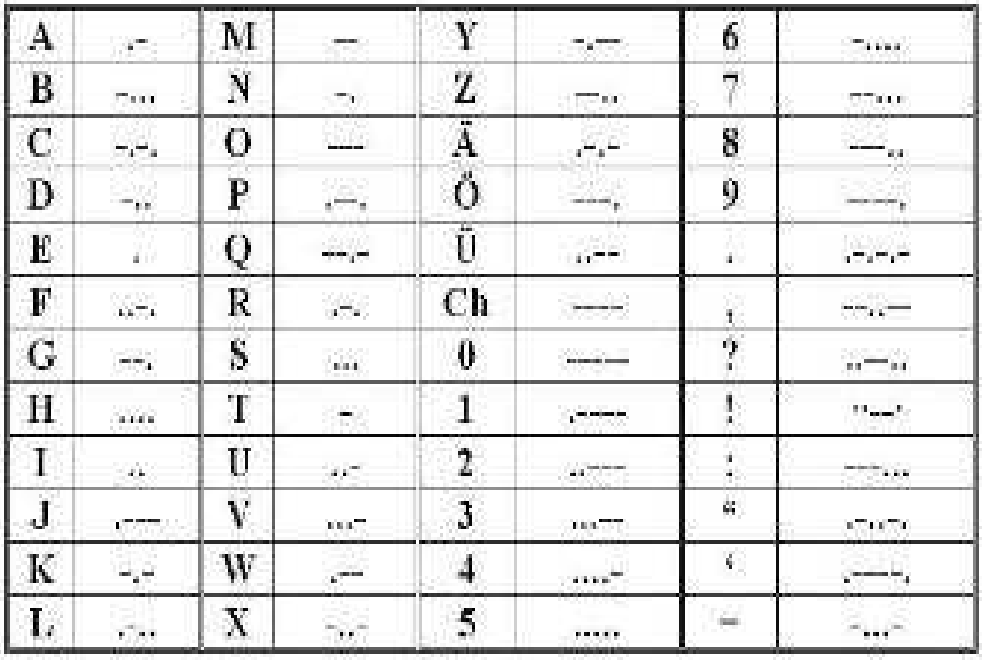
[1] *Disponível em:* http://operamundi.uol.com.br/conteudo/historia/29657/hoje+na+historia+1940+-+alemaes+comecam+a+utilizar+enigmas+na+franca+ocupada.shtml. *Acesso em 11 de junho de 2015.*

2.3 MÉTODOS CLÁSSICOS DE CRIPTOGRAFIA

Na seção anterior foi mencionado que é essencial para o sucesso de um processo criptográfico um código bem elaborado e de conhecimento apenas do remetente e do(s) destinatário(s) desejado(s). O termo código foi apresentado de forma generalizada e não-técnica mas para a criptologia, além do sistema de códigos, existe ainda um outro conceito envolvido na codificação das mensagens: as cifras.

Para codificar uma mensagem usando o *sistema de códigos* é imprescindível a introdução de um elemento chamado *livro de códigos*que deve ser compartilhado por remetente e destinatário. Este livro possui duas colunas. Em uma coluna temos a informação original (uma letra, palavra ou frase). Na outra temos esta mesma informaçãomas na forma codificada. Esta segunda será transmitida e o destinatário utilizando o mesmo livro que o remetente irá decodificá-la.

Na figura 1, Bruno e Fabiana compartilharam um livro de códigos para comunicarem-se. Neste livro cada letra corresponde a um número. Um exemplo famoso desta técnica é o *Código Morse* onde cada letra/número tem uma sequência codificada própria.



A	.-	M	--	Y	-.--	6	-....
B	-...	N	-.	Z	--..	7	--...
C	-.-.	O	---	Ä	.-.-	8	---..
D	-..	P	.--.	Ö	---.	9	----.
E	.	Q	--.-	Ü	..--	.	.-.-.-
F	..-.	R	.-.	Ch	----	,	--..--
G	--.	S	...	0	-----	?	..--..
H	....	T	-	1	.----	!	''--'
I	..	U	..-	2	..---	:	---...
J	.---	V	...-	3	...--	"	.-..-.
K	-.-	W	.--	4	....-	'	.----.
L	.-..	X	-..-	5	.....	-	-....-

Figura 4 – Código Morse

Fonte: Instituto Federal de Tecnologia do Rio Grande do Norte[2]

Contudo, os métodos criptográficos mais elaborados e utilizados utilizam-se de *cifras*. As cifras precisam de uma série de instruções também conhecidas como algoritmos. O algoritmo requer um elemento compartilhado conhecido como chave. Existe uma chave para codificar a informação e outra para decodificar(KHAN ACADEMY, [2---]). A codificação através de cifras apresenta-se com duas técnicas clássicas: as cifras de substituição e as cifras de transposição.

2.3.1 **Cifras de Transposição**

Nesta técnica os símbolos não são alterados. Troca-se a ordem dos símbolos de uma mensagem para uma outra ordem pré-determinada por um algoritmo. Para decodificar a mensagem deve-se realizar esta mesma trocamas na ordem inversa.

Um exemplo desta técnica é a transposição por colunas. Nela é utilizada uma chave onde não ocorre a repetição de caracteres. Cada caracter da chave é

colocado em uma coluna na primeira linha de uma matriz. Na segunda linha são numerados os índices da coluna colocando o número 1 abaixo do primeiro caracter da chave em ordem alfabética, o 2 abaixo do segundo caracter e assim sucessivamente. Após esta ordenação o texto a ser codificado é escrito horizontalmente. Utilizam-se quaisquer caracteres para completar uma linha caso a mensagem acabe antes da última coluna. O texto cifrado é escrito em colunas (em ordem crescente dos índices definidos anteriormente). Exemplo de codificação utilizando uma chave denominada EDU:

E	D	U
1	2	3
T	U	D
O	B	E
M	A	B

Texto claro: TUDOBEM

Texto cifrado: TOMUBADEB

2.3.2 **Cifras de Substituição**

Nesta técnica, ao contrário da anterior, os símbolos sofrem alteração se forem comparadas sua forma legível com sua forma codificada. O maior exemplo desta técnica são as *Cifras de César* onde é definido um valor de deslocamento (k) para o alfabeto, ou seja, cada símbolo original terá sua forma codificada igual o símbolo de posição n+k, onde n é a posição do símbolo original no alfabeto que está sendo utilizado. Exemplo:

k = 3.

Alfabeto:

1	2	3	4	5	6	7	8	9	10	11	12	13
A	B	C	D	E	F	G	H	I	J	K	L	M
14	15	16	17	18	19	20	21	#	23	24	25	26
N	O	P	Q	R	S	T	U	V	W	X	Y	Z

---

[2] *Disponível em: http://www.ebah.com.br/content/ABAAAe4KgAG/introducao-redes-computadores. Acesso em 11 de junho de 2015.*

Texto claro: TUDOBEM

Texto Codificado: WXGREP

2.4 CRIPTOGRAFIA COMPUTACIONAL

Nas seções anteriores foram apresentadas técnicas onde o homem utilizava lápis e papel para codificação/decodificação das mensagens. O surgimento de máquinas (como a Enigma) trouxe grande desenvolvimento para este campo de estudo. Mas foi o desenvolvimento computacional que trouxe uma revolução para a ciência da *Criptografia*. O processo de codificar mensagens é baseado em métodos matemáticos mas estes métodos, muitas das vezes, são tão complexos que inviabilizava o homem de pô-los em prática por não existir ferramenta que o ajudasse a implementá-los. O computador preencheu esta lacuna.

Na criptografia computacional existem basicamente 3 tipos de técnicas utilizadas: criptografia *Hash*, criptografia por Chaves Simétricas e criptografia por Chaves Assimétricas. Sendo que as mais utilizadas e detalhadas neste estudo são as duas últimas.

A criptografia *Hash* consiste em calcular para uma string (decomprimento variável) um valor digital de tamanho fixo (chamado de valor de *hash*). O algoritmo de *hash* busca gerar para cada string um valor de *hash* diferente. Sua segurança pode ser medida pela probabilidade de existirem strings diferentes com valores de *hash* iguais. Maior probabilidade menor segurança, menor probabilidade maior segurança. Ou seja, mesmo que a *string* mude apenas um bit em comparação com uma outra qualquer o valor de *hash* deve ser diferente. Matematicamente a função *hash* pode ser descrita como H(x), onde H(x) é uma função que não permite,

computacionalmente, fazer o sentido inverso, ou seja, calcular a *string* x conhecendo o valor do *hash.*

Intuitivamente, uma função de *hash* calcula, rápida, segura e univocamente, representantes adequadamente curtos (chamados resumos) para mensagens arbitrariamente longas. Esses resumos são assinados em lugar das próprias mensagens, mantendo em nível aceitável o esforço computacional comumente encontrado durante a operação de algoritmos assimétricos (BARRETO,2003)

Outra técnica existente é a criptografia de chaves simétricas. Esta técnica utiliza um algoritmo que utiliza a mesma chave para codificar e decodificar as mensagens. Se uma pessoa quer se comunicar com outra com segurança, ela deve passar primeiramente a chave utilizada para cifrar a mensagem. Este processo é chamado distribuição de chaves, e como a chave é o principal elemento de segurança para o algoritmo, ela deve ser transmitida por um meio seguro (TRINTA, MACÊDO, 1998). É o tipo mais simples de criptografia, tanto do ponto de vista de processo, quanto do computacional. Esta técnica será detalhada no próximo capítulo.

Finalmente existe a técnica de criptografia de chaves assimétricas. Algoritmos que utilizam esta técnica, precisam de duas chaves diferentes, uma para codificar e outra para decodificar as mensagens. É uma técnica que envolve um processo mais complexo para criação e troca das chaves além de exigir mais recursos computacionais para implementá-la.

Esse par de chaves pode ser visto como relativamente independente entre si. Esse atributo de "independência relativa" pretende significar que não deve ser possível deduzir-se uma chave a partir do conhecimento da outra (GUIMARÃES, 2001). Este tipo de criptografia será detalhado no capítulo 4.

Chaves criptográficas são um conjunto de bits baseados em um algarismo capaz de interpretar a informação, ou seja, capaz de codificar e decodificar. Se a chave do receptor não for compatível com a do emissor, a informação então não será extraída (SEGURANÇA DA INFORMAÇÃO, 2---).

Na criptografia computacional a segurança do algoritmo que esteja sendo usado é proporcional ao tamanho da chave de criptografia. Esta chave tem o tamanho medido em bits. Assim, um algoritmo que use uma chave de 8 bits pode ter $2^8 = 256$ chaves diferentes. Já um algoritmo de 128 bits teria 2^{128}chaves diferentes o que dificultaria muito a descoberta desta chave mesmo utilizando-se um computador.

Nos próximos capítulos serão detalhados os principais algoritmos de criptografia, simétricos e assimétricos, as características de cada um e suas vantagens e eventuais fragilidades.

3 **CRIPTOGRAFIA SIMÉTRICA**

É o tipo de criptografia computacional mais simples que existe já que a mesma chave será utilizada para codificar e decodificar as mensagens. Para isto é necessário que tanto o remetente quanto o destinatário tenham prévio conhecimento desta chave.

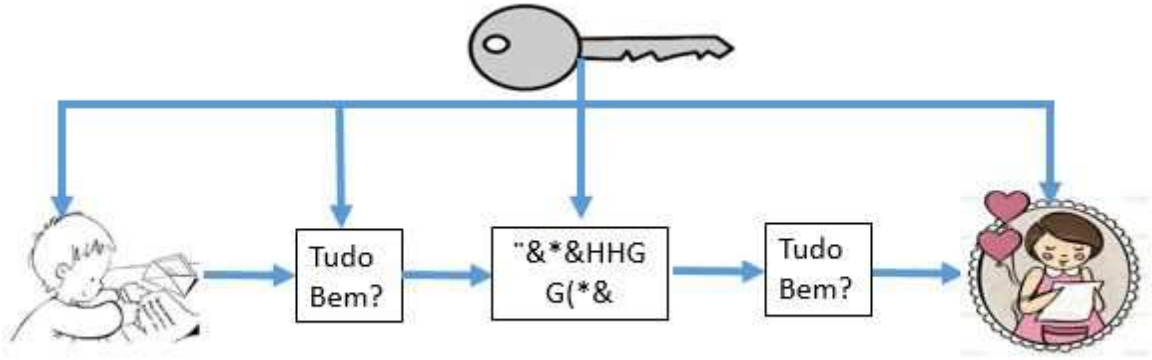


Figura 5 – Processo de Criptografia Simétrica

A maior vantagem dos algoritmos simétricos, em relação aos assimétricos, por ter a presença de uma chave única, é a sua execução mais rápida, por ser menos complexo, exigindo menor capacidade computacional do hardware que os utilizará. Uma preocupação constante de quem utiliza estes algoritmos é a maneira como a chave secreta é compartilhada. Se a mesma for enviada por um meio não seguro ela estará sujeita à interceptação podendo comprometer toda a segurança do processo. Por isto, na maioria das vezes, os dois métodos são executados concomitantemente: para troca de chaves é utilizado um algoritmo assimétrico e, após a troca, é utilizado um algoritmo simétrico para codificação da mensagem.

Um algoritmo simétrico tem sua segurança avaliada pelo tamanho máximo (em bits) da chaveque ele utiliza. Assim, um algoritmo que utiliza-se de uma chave de 128 bits pode ser considerado mais seguro, de uma maneira geral, do que outro que utiliza-se de uma chave de 32 bits. O quadro abaixo avalia, para um computador

hipotético, o tempo que um algoritmo de força bruta levaria para decifrar uma determinada chave de criptografia baseada em seu tamanho.

Tamanho da chave(bits)	Combinações possíveis	Tempo para quebra (1000 Chaves geradas/ps)
64	2^{64}	5,1 horas
128	2^{128}	10^{16} anos
256	2^{256}	10^{51} anos

Figura 6 – Tempo para Descoberta de uma chave Criptográfica

Do exemplo acima pode-se notar a importância do tamanho da chave para um algoritmo criptográfico. Deve-se atentar, contudo, para o fato de que conforme maior for a chave, maior será a capacidade computacional exigida para o algoritmo. Será necessário avaliar qual o valor da informação que está sendo protegida e se o custo do algoritmo é viável para a proteção de tal informação. Informações mais valiosas exigirão algoritmos mais caros (com chaves maiores) informações menos valiosas o inverso.

São diversos os algoritmos simétricos existentes. Dentre os principais destacam-se o DES(*Data Encryption Standard* - detalhado a seguir), o 3DES (*Triple Data Encryption Standard*) e o AES (*Advanced Encryption Standard*).

3.1 ALGORITMO DES

Na próxima subseção será apresentado um breve resumo do surgimento do DES (explicando as necessidades que levaram a sua criação) e porque ele pode ser cosiderado “o pai” de todos os outros algoritmos que o sucederam. Na subseção posterior será detalhado o seu funcionamento através de explicações teóricas e um exemplo prático da criptografia de uma pequena mensagem.

3.1.1 **Histórico**

O DES foi um dos primeiros algoritmos desenvolvidos com o objetivo de criptografar informações. Na década de 70 a IBM, já prevendo o exponencial aumento da comunicação digital, desenvolveu um algoritmo chamado LÚCIFER, que foi a base do DES. O DES foi desenvolvido tentando eliminar as fraquezas do LUCIFER. Antevendo a importância estratégica deste trabalho, a NSA *(National Security Agency*) absorveu o algoritmo tornando-o secreto.

Em 1976 ele foi considerado uma norma federal americana. Em 1977 ele foi autorizado a se tornar um algoritmo público rapidamente tornando-se um padrão de mercado. Muitos tinham receio de usá-lo pios existiam dúvidas de que de que o algoritmo possuía um *backdoor* desenvolvido pela própria NSA para fins de espionagem. Apesar da desconfiança, até hoje, nada foi provado e o DES é considerado o mais importante algoritmo de criptografia já criado pois foi responsável pelo desenvolvimento da criptografia para fins não militares, fato até então inédito.

3.1.2 **Funcionamento**

O DES é um algoritmo de cifragem em blocos: ele transforma um bloco de entrada (em texto claro) de 64 bits em um texto codificado também de 64 bits. Para isto ele utiliza uma chave de codificação de 56 bits mais 6 bits de paridade (total também de 64 bits). No início o tamanho da chave de 56 bits foi considerado pequeno, surgindo uma outra desconfiança de que o motivo para a NSA ter adotado este tamanho é a possibilidade dela poder quebrar uma chave através de força bruta antes de outros postulantes, devido ao grande poder computacional que ela possui. Apesar disto, somente em 1998 (20 anos depois de tornar-se público) o algoritmo foi quebrado através do método de força bruta.

De uma maneira geral, o fluxo do DES pode ser ilustrado como:

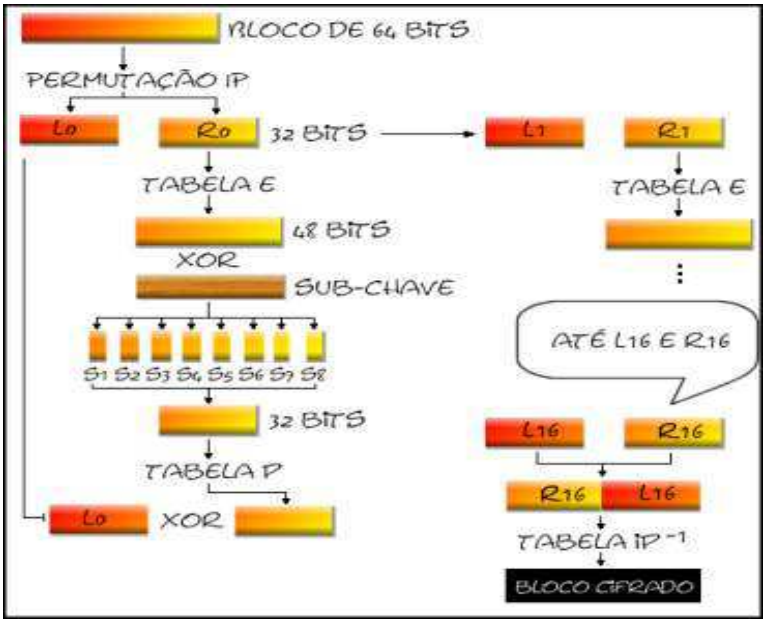


Figura 7 – Fluxo do algoritmo DES.

Fonte: Aldeia Numaboa[3]

Para descrição do processo de funcionamento do DES, será considerada a seguinte mensagem em texto claro: “Tudo Bem” e a chave simétrica será “vascaino”.

Assim devemos seguir os seguintes passos para codificação da mensagem acima com a chave apresentada:

Permutação IP: o 58º bit de M passa ser o 1º bit de IP, o 50º bit será o 2º e assim sucessivamente, de acordo com a próxima tabela:

[3] *Disponível em: http://www.numaboa.com.br/criptografia/bloco/313-des2?showall=1&limitstart=. Acesso em 04 de novembro de 2015.*

Tabela 1 – Permutação a partir de uma cadeia inicial M

IP							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

M = 0101 0100 0111 0101 0110 0100 0110 1111 0010 0000 0100 0010 0110 0101 0110 1101 (Este é o valor da string “Tudo bem” em binário).

IP = 1110 1111 0000 0011 11101111 1100 1010 00000000 10011110 1000 1000 0010 0100 (É a cadeia M acima permutada de acordo com a tabela apresentada).

- Lo: 1ª metade (32 bits iniciais de IP) = 1110 1111 0000 0011 1110 1111 1100 1010
- Ro: 2ª metade (32 bits finais de IP) = 0000 0000 1001 1110 1000 1000 0010 0100

Prossegue-se então com 16 iterações (1<=n<=16), usando-se as seguintes fórmulas para o cálculo de Ln e Rn:

$L_n = R_{n-1}$

Rn = Ln-1 + f(Rn-1,Kn), onde f é uma função que opera com 2 blocos: Rn-1 (32 bits) e K_n (48 bits).

K é a chave de codificação do processo. Conforme informado acima a chave é a palavra “vascaino” que transformando para sua notação binária fica:

K = 011101100110000101110011011000110110000101101001011011100110111 (64 bits).

Apesar do tamanho inicial da chave ser 64 bits ela sofre transformação no decorrer do processo conforme descrito a seguir:

- $K_{n:}$ é uma subchave da chave total de 56 bits:

a) Cálculo das 16 subchaves de 48 bits: a chave principal de 64 bits é permutada de acordo com a seguinte tabela:

Tabela 2 – Permutação a partir de uma chave inicial de 64 bits

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

. Observa-se que são permutados apenas 56 bits da chave total. Isto ocorre porque o último bit de cada caracter é considerado um bit de paridade (bits 8, 16, 24, 32, 40, 48 56 e 64) e, por isso, não são considerados no processo de permutação.

Considerando K como a chave simétrica na sua forma binária:

K = 01110110011000010111001101100011011000010110100101101110011011111, obtêm-se a seguinte chave permutada de 56 bits:

K+ = 10000000111111111111111100001100110111100001111000101

Esta chave K+ é desmembrada em duas chaves diferentes, C_0 (1ª metade de K+) e D_0 (2ª metade de K+), cada uma com tamanho de 28 bits. Assim, teremos:

Co = 1000000 0111111 1111111 1110000

Do = 1100110 1111000 0111100 0000101

Para o cálculo dos Cn e Dn, até 1<=n<=16, é considerado o par anterior $C_{n-1,}$ D_{n-1}com deslocamento de bits à esquerda de acordo com a tabela seguinte:

Tabela 3: Deslocamento à esquerda de bits até n = 16

n	Deslocamento à esquerda
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	2
10	2
11	2
12	2
13	2
14	2
15	2
16	1

Assim, partindo de Co e Do podemos obter Cn,Dn até n = 16:

Tabela 4: Cálculo de Cn,Dn

Co = 10000000111111111111111110000 Do = 11001101111000011111000000101
C1 = 00000001111111111111111100001 D1 = 10011011110000111110000001011
C2 = 00000011111111111111111000010 D2 = 00110111100001111100000010111
C3 = 00001111111111111111100001000 D3 = 11011110000111110000001011100
C4 = 00111111111111111110000100000 D4 = 01111000011110000000101110011
C5 = 11111111111111111000010000000 D5 = 11100001111000000010111001101
C6 = 11111111111111100001000000011 D6 = 10000111100000001011100110111
C7 = 11111111111100001000000001111 D7 = 00011110000001011100110111110
C8 = 11111111110000100000000111111 D8 = 01111000000101110011011110000
C9 = 11111111000010000000011111111 D9 = 11100 00001011100110111100001
C10 = 11111100001000000011111111111 D10 = 100 00001011100110111110000111
C11 = 11110000100000001111111111111 D11 = 00000101110011011111000011110
C12 = 11000010000000111111111111111 D12 = 00010111001101111100001111000
C13 = 00001000000011111111111111111 D13 = 01011100110111110000111100000
C14 = 00100000001111111111111111100 D14 = 01110011011110000111100000001
C15 = 10000000111111111111111110000 D15 = 11001101111000011110000000101
C16 = 00000011111111111111111000010 D16 = 00110111100001111100000010111

b) Cálculo de Kn: Para 1<=n<=16 podemos achar K_n aplicando permutação sobre a concatenação de C_n,D_n considerando a seguinte tabela:

Tabela 5: Permutação de Cn,Dn para cálculo de Kn

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Nota-se que apesar da concatenação possuir 56 bits, são considerados apenas 48 bits para Kn. Então, após a permutação os valores de Kn ficariam:

Tabela 6: Cálculo de K_n

K_1	111000011011111001100110000101000110100101100111
K_2	111000001011011001111110011101000100100110010101
K_3	111001001101011001110110110000110010000011011011
K_4	111001101101101101110010111001111011001100001001
K_5	101011101111001101110011001100100001011101101110
K_6	101011110101011101011011010111001001100110100110
K_7	0010 111010100111101101111100000111111001100000 11
K_8	000111111101100111011001011010111011100001011001
K_9	000111110110100111011011101000111101010100111010
K_{10}	001111110110110110001101000011010001111100100110
K_{11}	010110110010110110001101110111000100100011110100
K_{12}	010110011010110010111101010000011100101011011101
K_{13}	110101011010110010101110100100111011010010011001
K_{14}	111100101010111010100110101010110001011100100101
K_{15}	111110001011111000100110000110100110101110100110
K_{16}	111000001011011001111110011101000100100110010101

Função f(Rn-1,Kn): Conforme visto no passo a) para o cálculo de Rn deve-se fazer um XOR (+) entre Ln-1 e f(Rn-1,Kn). Para n=1 temos:

$L_{1-1} = L_0 = 11101111000000111110111111001010$

A função f consiste em um XOR entre Rn-1 e Kn. Mas estas duas variáveis possuem tamanho distintos. Enquanto a primeira tem tamanho de 32 bits a segunda possui 48 bits. Para o cálculo do XOR deve-se expandir R para 48 bits também.

Nesta expansão é considerada a seguinte tabela de permutação, onde alguns bits de R são repetidos:

Tabela 7: Expansão de R para 48 bits

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
19	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29

Logo Rn-1 = R0 = 0000.0000.1001.1110.1000.1000.0010.0100 e R0 expandido usando-se a tabela acima fica: 000000.000001.010011.111101.010001.010000.000100.001000. Ou seja, de cada bloco original de 4 bits são produzidos blocos expandidos de 6 bits.

Após a expansão pode-se realizar o XOR entre K,R pois agora as duas variáveis possuem o mesmo comprimento: 48 bits. Assim, f(Rn-1,K1) fica: 111000.011010.101010.011011.010100.010110.100001.101111.

Após o XOR acima foram criadas 8 cadeias de 6 caracteres cada. Cada cadeia será o parâmetro de entrada para uma caixa “S” (ou S boxes) que é uma caixa de seleção e um dos componentes chave do algoritmo DES. A 1ª cadeia é o parâmetro da caixa 1, a 2ª da caixa 2 e assim sucessivamente. O objetivo de cada caixa é receber uma cadeia de 6 caracteres e produzir uma cadeia de apenas 4.

Tomando como exemplo a caixa S1, ela possui a seguinte função de seleção:

Tabela 8: Função de Seleção S1

Função S1															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
|	----------															
0 |	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1 |	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2 |	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3 |	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

A função receberá uma cadeia B de 6 bits e separará esta cadeia em duas subcadeias: a 1ª conterá o 1º e o último bit da cadeia original. A 2ª conterá os bits 2, 3, 4 e 5. A s duas subcadeias transformarão seus bits em valores decimais. O valor encontrado na 1ª cadeia será a linha a ser utilizada na função e o valor da 2ª será a coluna da função. De posse da linha e coluna retorna-se um valor decimal. Este valor será transformado novamente em uma cadeia binária de 4 dígitos, sendo o resultado retornado pela função.

Ex.: o 1º bloco de caracteres de F é **111000**. A 1ª subcadeia recebe o valor 10 (1º e último bit) que em decimal corresponde a 2 (esta é a linha da função). A 2ª subcadeia recebe o valor 1100 (2º, 3º, 4º e 5º bit) que em decimal corresponde a 12 (esta é a coluna da função). Assim, procurando-se na função a linha 2, coluna 12, acha-se o decimal 7 que transformado para binário é o valor **1101**. Esta é a saída da função S1.

O mesmo processo é seguindo por todas as caixas de seleção, sendo que cada uma tem a sua própria função e cada função retorna valores diferentes dependendo da linha/coluna desejada.

Após todas as caixas de seleção retornarem seus resultados obtêm-se f = 1101.0000.0011.0111.0011.0000.0110.1101. Agora pode-se calcular $R_1 = L_0 + f(R_0, K_1)$:

L_0 =1110 1111 0000 0011 1110 1111 1100 1010

$f(R_0,K_1)$ = 1101.0000.0011.0111.0011.0000.0110.1101

R_1 = 0011.1111.0011.0100.1101.1111.1010.0111

O mesmo processo é seguido até achar-se L_{16},R_{16}. Ao final resulta-se num bloco de 64 bits. Este bloco é invertido para uma cadeia R_{16},L_{16}. Aplicando-se então uma permutação final de acordo com a seguinte tabela:

Tabela 9: Permutação final de R_{16},L_{16}

IP^{-1}

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

L_{16} = 10000010110010100000101100100001

R_{16} = 11100000110110001111111000110000

L_{16},R_{16}=

10000010110010100000101100100001111000001101100011111110001100 00

R_{16},L_{16}=

11100000110110001111111000110000100000101100101000001011001000 01

$(R_{16},L_{16})^{-1}$=

00001010101011000000010000111100000101010100011101110100111101 00

Esta última permutação é o resultado final.

Mensagem original:

Binário:

0101010001110101011001000110111100100000010000100110010101101101

String: Tudo bem

Mensagem codificada:

Binário:

0000101010101100000001000011110000010101010001110111010011110100

String: ?<Gt?

O processo de decifrar a mensagem é basicamente o mesmo, lembrando-se que a mesma chave que codificou será a responsável pela decodificação já que está sendo utilizado um algoritmo de codificação simétrico.

O algoritmo 3DES é uma evolução do DES e surgiu da necessidade de eliminar-se algumas fragilidades do DES. Ele trabalha com chaves de 112 ou 168 bits (DES = 56 bits) e possui um total de 48 iterações (DES = 16 iterações).

O DES triplo utiliza o sistema EDE (Encrypt-Decrypt-Encrypt) para cifragem, ou seja, o texto plano é encriptado primeiro com a primeira chave, decriptado com a segunda e encriptado novamente com a terceira. O efeito geral é o de haver-se cifrado com uma chave de tamanho duplo e por tanto muitíssimo mais seguro. As chaves são obtidas utilizando-se o mesmo método descrito no capítulo anterior (DES) (MORAES,2004).

O AES foi algoritmo criado por Vincent Rijmen e Joan Daemen para concorrer (e vencer) um concurso proposto pelo NIST (*National Institute of Standards and Technology* dos EUA). Para participar do concurso, o algoritmo precisaria preencher alguns requisitos: ser um algoritmo público, utilizar codificação em blocos, projetado

para tamanho de chaves crescentes, implementável em hardware e software, dentre outros.

Este algoritmo trabalha em blocos de dados de tamanho fixo de 128 bits, chamados de State, os quais são organizados como uma matriz de quatro linhas e quatro colunas de bytes. Os tamanhos das chaves podem ser três: 128 bits, 192 bits, ou 256 bits (STALLINGS, 2008 apud PIGATTO, 2012)

4 **CRIPTOGRAFIA ASSIMÉTRICA**

Enquanto no processo anterior utiliza-se a mesma chave para codificação e decodificação das mensagens (chave simétrica), na criptografia assimétrica (também chamada de criptografia de chave pública) utilizamos duas chaves diferentes: uma para codificar (chave pública) e outra para decodificar (chave privada). O destinatário é o responsável pela geração de ambas as chaves e pela divulgação da chave pública. A chave pública é de conhecimento de um ou vários remetentes e a chave privada pertence apenas ao destinatário. O processo, simplificadamente, pode ser descrito conforme a figura abaixo:

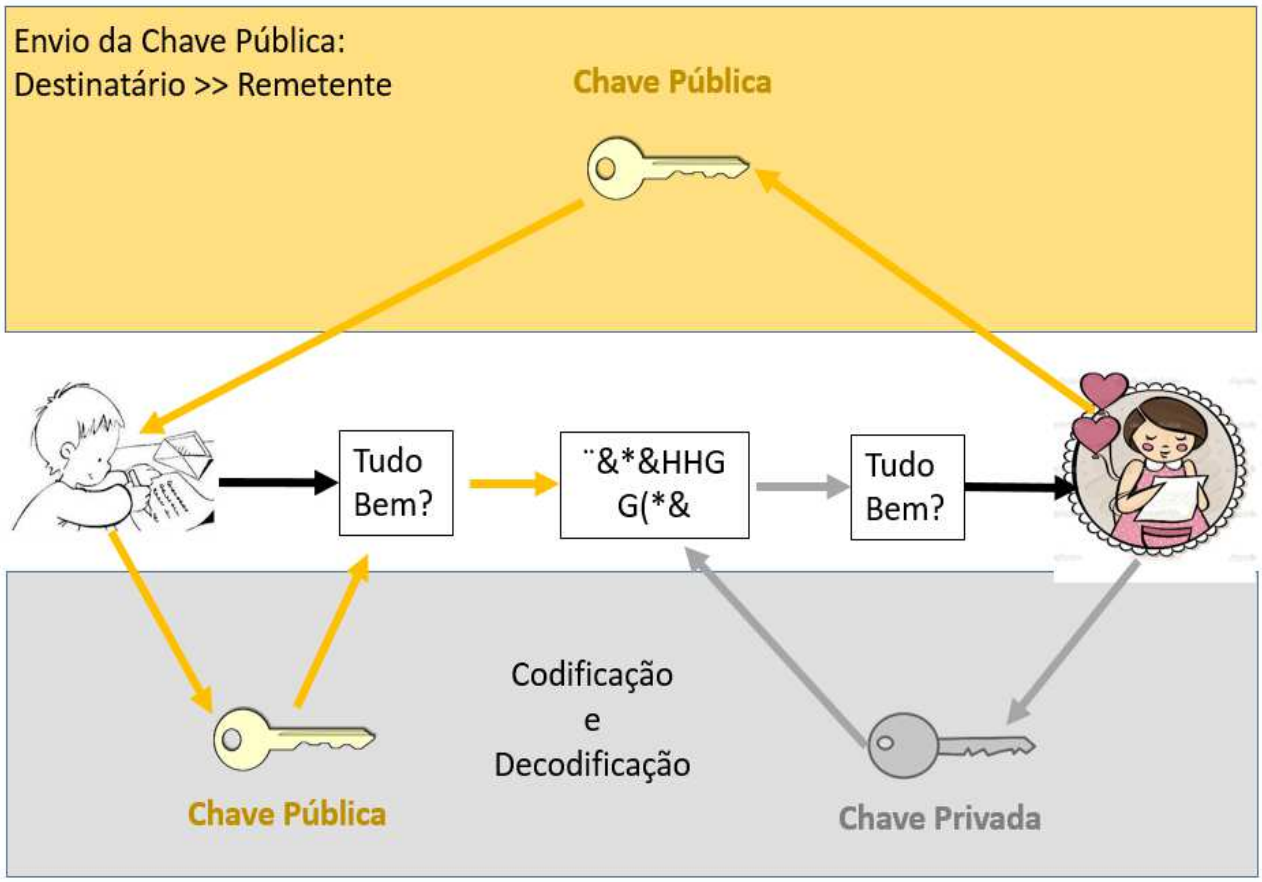


Figura 8 – Codificação e Decodificação com Criptografia Assimétrica

Detalhando o processo ilustrado:

i – Fabiana gera um par de chaves pública/privada ;

ii – Fabiana (a destinatária) envia sua chave pública (responsável pela codificação) para Bruno (o remetente);

iii – De posse da chave, Bruno a utiliza para a codificação da mensagem “Tudo Bem?”. A mensagem codificada é enviada para Fabiana.

iv – Fabiana então utiliza sua chave privada para decodificar a mensagem.

Um algoritmo assimétrico deve ser capaz de gerar um par de chaves de tal forma que de posse de apenas uma delas a outra não pode ser deduzida, ou seja, que possui a chave pública não deve ter a capacidade computacional de descobrir a privada.

Anteriormente foi visto que no algoritmo simétrico a chave ao ser enviada em um canal não-seguro está sujeita à interceptação. Caso o interceptador consiga decifrá-la, esta mesma chave que codificou a mensagem será a responsável por decodificá-la expondo uma fragilidade na comunicação. O algoritmo assimétrico resolveu este problema. A chave privada não trafega por um canal público. Assim, somente a chave pública está sujeita à interceptação e, uma vez obtida por terceiros não-autorizados, não terá a capacidade de decifrar mensagens criptografadas neste processo (apenas a privada terá esta capacidade).

Apesar de apresentar maior segurança em comparação com algoritmos de chave simétrica, a criptografia assimétrica possui uma desvantagem em relação ao método anterior: grande lentidão para codificar e decodificar uma mensagem (por exigir grande capacidade computacional).

Como cada método tem as suas vantagens e desvantagens, na maioria dos processos de criptografia computacionais, eles são utilizados em conjunto. Utiliza-se

a criptografia assimétrica para a troca das chaves simétricas (trazendo maior segurança ao processo) e após a troca de chaves entre remetente/destinatário a codificação da mensagem é feita utilizando-se a criptografia simétrica (trazendo maior velocidade). O processo pode ser ilustrado na figura a seguir:

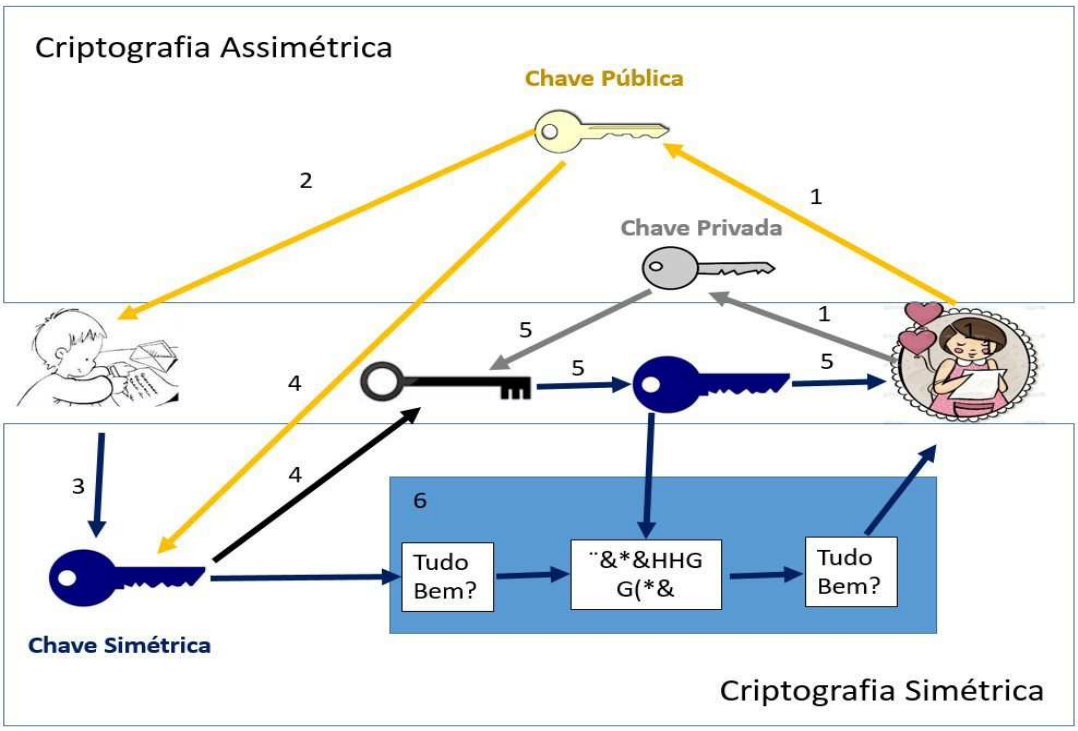


Figura 9 – Troca de Chaves e Envio de Mensagens Criptografadas

Detalhando a figura acima descreve-se o processo.

1 – Fabiana gera um par de chaves pública/privada (criptografia assimétrica);

2 – Fabiana envia a chave pública para Bruno;

3 – Bruno gera sua chave simétrica;

4 – Bruno codifica sua chave simétrica com a chave pública enviada por Fabiana. Bruno sua chave simétrica codificada para Fabiana;

5 – Fabiana recebe a chave simétrica codificada de Bruno e decodifica-a utilizando sua chave privada;

6 – Bruno codifica suas mensagens com a chave simétrica e envia para Fabiana, que de posse da mesma chave simétrica decodifica a mensagem.

Dentre os principais algoritmos de criptografia assimétrica utilizados destacam-se: RSA e Diffie-Helman. Este último foi o responsável introduzir o conceito de chave pública em 1976 e possui o nome de seus criadores Whitfield Diffie e Martin Hellman. O RSA é o algoritmo assimétrico mais utilizado atualmente e será detalhado na próxima seção.

4.1 ALGORITMO RSA

Na próxima subseção será apresentado um breve histórico sobre o surgimento do algoritmo e sobre os seus criadores e posteriormente será detalhado seu funcionamento através de explanações teóricas. A seção é finalizada através da criptografia de uma pequena mensagem utilizando o RSA.

4.1.1 Histórico

Apesar de Diffie-Hellman serem considerados os criadores do conceito de chave pública, Ron Rivest, Adi Shamir e Leonard Adleman são os responsáveis pela popularização da utilização da criptografia assimétrica. Os 3 pesquisadores do MIT *(Massachusetts Institute of Technology*), cuja inicial de cada sobrenome denomina o algoritmo, criaram o RSA em 1978, implementando a mais perfeita criptografia de chave pública. O RSA foi criado a partir da idéia de Diffie-Helman. Rivest leu o trabalho dos percursores e procurou uma função matemática capaz de transformar a idéia original em um sistema de criptografia real e viável. (Aldeia Numaboa, 2005). Shamir juntou-se ao colega na busca desta função.

Sempre que uma função candidata era descoberta os 2 pesquisadores pediam a ajuda de Adleman que procurava por erros no conceito apresentado. Após várias tentativas frustradas finalmente Rivest e Shamir acharam uma função que não

permitia nenhuma contestação por parte de Adleman. Assim estava criada a primeira cifra assimétrica perfeitamente implementada.

4.1.2 Funcionamento

O principal conceito matemático básico utilizado pelo algoritmo RSA são os números primos. Números primos são aqueles divisíveis apenas por 1 e por ele mesmo, excetuando-se o algoritmo 1 (ex.: 3, 5, 7, 11, etc). Outro conceito é o de números primos entre si, ou seja, 2 números possuem como divisor comum apenas o número 1.

Exemplo: 8 e 9 são primos entre si. Divisores de 8 = 1,2. Divisores de 9 = 1,3. O único divisor comum é o 1.

Outro conceito é a aritmética modular onde não é considerado um conjunto infinito de números mas um conjunto finito que será igual ao módulo a ser utilizado. Se definirmos um módulo igual a 6 o conjunto a ser considerado variará de [0,6]. Exemplo: a operação 5 x 3 utilizando o módulo 6 será igual a 5x3 = 3 (mod 6). Para achar-se o resultado divide-se o produto dos fatores à esquerda (15) pelo módulo. O resto desta divisão é o resultado da operação (3) no módulo desejado (6).

Rivest, Shamir e Adleman criaram um função unidirecional, ou seja, que não permite ser revertida para seus valores iniciais em sua teoria. Na prática esta função pode ser revertida mas seu custo é tão alto que não compensa o esforço que será dispendido na reversão.O algoritmo proposto pelos 3 pesquisadores pode ser dividido em 5 fases descritas a seguir. Para efeito de ilustração será considerado a mesma mensagem anterior (Tudo Bem).

4.1.2.1 Escolha de 2 números primos

O responsável pela geração do par de chaves pública/privada deve escolher 2 números primos (preferencialmente grandes) chamados de *p* e *q*. Aqui se encontra

um paradigma: quanto maior os números primos, maior será a segurança e a lentidão para o processamento do algoritmo. Assim, na implementação do algoritmo, deve ser levada em consideração a relação segurança x desempenho. Para este exemplo, a efeito de ilustração, serão escolhidos os números primos 23 e 41.

4.1.2.2 Cálculo da chave pública (e,n)

A partir de p = 23 e q = 41 acha-se n = p x q. Assim, n = 23 x 41 = 943. Agora se escolhe um número *e* que seja primo entre si com (p-1)(q-1) = 880. 880 = 2 x 2 x 2 x 2 x 5 x 11. O número *e* deve ser um número que não seja divisível por nenhum divisor de 880, ou seja, 2, 5 e 11. Pode ser considerado *e* = 3. Então n = 943 e *e=3*é o par da chave pública.

4.1.2.3 Cálculo de chave privada (d,n)

O inverso de um número x é 1/x. O inverso de 3 é 1/3. Para o cálculo da chave privada no RSA deve-se utilizar uma função chamada de redução modular. Para exemplificar considera-se a expressão modular 5 (mod 9) onde o resultado é igual a 5. Para encontrar a inversão desta expressão (ou sua redução modular) deve-se achar um número que multiplicando 3 resulte 1 no módulo 9. Fazendo por tentativa e erro ficaria:

5 x 1 (mod 9) = 5

5x 2(mod 9) = 1 << 2 é a redução modular da expressão 5 (mod 9)

Generalizando este cálculo precisa-se achar uma redução modular através da seguinte expressão: *e* x *d* (mod (p-1)(q-1)) = 1, onde *d* é a parte a ser encontrada da chave privada.No exemplo deve-se achar a redução 3 x d (mod 880). O cálculo por tentativa e erro acima é muito demorado, assim o método RSA implementa o *algoritmo de Euclides estendido (*algoritmo que retorna uma combinação linear na forma de ax + by = MDC(a,b))*:*

Neste exemplo o algoritmo calcularia e x d - ((p-1)(q-1) x b = MDC(e,(p-1)(q-1) ou 3d – 880b = MDC(3,880) = 1. O algoritmo estendido usa como base os restos diferentes de zero encontrados pelo algoritmo padrão de Euclides (utilizado para achar o MDC entre dois números naturais). Após o cômputo do algoritmo estendido, acha-se d=587. Assim, foi gerado o par de chaves privadas (d,n) = (587,943).

Obs.: A implementação dos algoritmos serão apresentados no anexo B deste trabalho.

4.1.2.4 Publicação da chave pública

Após a definição das chaves,o destinatário pode enviar para seus remetentes a chave pública que será utilizada para a codificação das mensagens. Deve-se tomar cuidado particular com a confidencialidade de seu par de chaves privada, caso contrário todo o processo de segurança estará comprometido, pois este par de chaves será o responsável pela a decodificação da mensagem.

4.1.2.5 Codificação das mensagens

Após o envio do par de chaves pública o processo de codificação pode ser iniciado. O primeiro passo é transformar a mensagem a ser codificada em caracteres ASCII:

T = 84

u = 117

d = 100

o =111

“Espaço” = 32

B = 66

e = 101

m = 109

Assim a mensagem “Tudo Bem” representada por seus caracters ASCII seria substituída pela cadeia m = 841171001113266101109. Agora a cadeia m deve ser separada em blocos de tamanho menor ou igual ao número de dígitos de n. Como n = 943, m pode ser dividido em blocos de 3 caracteres cada. Então:

m_1= 841

m_2= 171

m_3= 001

m_4= 113

m_5=266

m_6= 101

m_7=109

A mensagem então é cifrada em blocos c_i através da fórmula $c_i=m^e(\text{mod } n)$.

$c_1=841^3(\text{mod } 943) = 610$

$c_2 =171^3(\text{mod } 943) = 425$

$c_3 =001^3(\text{mod } 943) = 001$

$c_4 =113^3(\text{mod } 943) = 107$

$c_5 =266^3(\text{mod } 943) = 702$

$c_6 =101^3(\text{mod } 943) = 545$

$c_7 =109^3(\text{mod } 943) = 290$

Assim a mensagem cifrada é:610425001107702545290.

4.1.2.6 Decodificação da mensagem

Para a decodificação da mensagem anterior utiliza-se a mesma fórmula de codificação, apenas com uma alteração: troca-se o expoente *e* (chave pública) pelo expoente *d* (chave privada). Assim a fórmula fica: $m_i=c^d(\text{mod } n)$.

$m_1=610^{587}(\text{mod } 943) = 841$

$m_2 = 425^{587} (\text{mod } 943) = 171$

$m_3 = 001^{587} (\text{mod } 943) = 001$

$m_4 = 107^{587} (\text{mod } 943) = 113$

$m_5 = 702^{587} (\text{mod } 943) = 266$

$m_6 = 545^{3} (\text{mod } 943) = 101$

$m_7 = 290^{3} (\text{mod } 943) = 109$

Ao final dos cálculos é gerada a mensagem original na sua forma ASC II: 841171001113266101109.

5 **PROPOSTA DE AUTENTICAÇÃO BASEADA NO CADASTRO DO USUÁRIO**

Visando proporcionar maior segurança no acesso de clientes ao site de uma instituição financeira, este capítulo apresentará um estudo de caso onde será feito a autenticação destes clientes no momento da digitação dos seus dados de acesso (agência, conta e senha) à base de dados da instituição. Será utilizada a técnica de criptografia assimétrica vista no capítulo anterior onde a geração das chaves privada e pública será de responsabilidade de um programa instalado no servidor da instituição financeira. O critério para geração destas chaves será qualquer informação pessoal do usuário, escolhida de forma aleatória, cadastrada na base de dados da empresa.

Para a proposta descrita no parágrafo anterior será desenvolvido um software através da linguagem de programação Java e que será executado na plataforma móvel Android. O código do software será apresentado no anexo deste estudo. As telas do software serão mostradas nesta seção.

5.1 VANTAGENS

A necessidade desta criptografia efetuada pela aplicação, além daquelas já presentes na pilha de protocolos TCP/IP utilizadas pelos usuários, é certificar que um método de segurança proprietário, não conhecido por potenciais hackers, será utilizado para o transporte destas informações confidenciais. Outra questão é a utilização da aplicação de home banking em um ambiente não seguro (como redes abertas), onde esta solução proprietária será a única forma de segurança que garantirá a comunicação usuário-instituição.

Outra vantagem da implementação desta solução é uma dupla proteção caso o usuário já esteja navegando por um ambiente seguro. Aqui a criptografia é feita na

camada de aplicação e a informação ainda será criptografada nas camadas inferiores (SSL, por exemplo). Caso um hacker descubra uma vulnerabilidade de um protocolo conhecido descriptografando as mensagens deste protocolo, a informação ainda estará criptografada pela solução-proprietária.

5.2 PREMISSAS DO SOFTWARE

No momento da abertura de uma conta corrente o usuário efetua um cadastro junto à instituição financeira onde são informadas várias questões de cunho pessoal, tais como: nome da mãe, pai, data de nascimento, local de nascimento, endereço, telefones, entre outras. Estas informações pessoais serão escolhidas pelo software, de forma aleatória, a cada momento que precede a digitação da senha. A partir desta escolha, serão gerados os números primos que servem de base para o processo da criptografia assimétrica. Desta maneira, cada vez que a senha for digitada, ela será criptografada por uma chave pública diferente tendo como resultado um resultado criptográfico diferente. Assim, caso a conexão sofra um ataque *man-in-the-middle*, o potencial intruso terá a impressão que o sistema implementa o método OTP (*one-time-password*) e que, caso ele venha a descobrir esta senha, ela não terá valor algum visto que no método OTP uma senha só pode ser utilizada uma única vez e então é descartada. Tal fato pode desencorajar ataques a este tipo de conexão.

5.3 DESENVOLVIMENTO DO SOFTWARE

A seguir encontra-se a tela principal do software que irá demonstrar o esquema proposto:

Figura 10 – Tela Principal do Sistema MotBank – versão usuário

Na tela principal mostrada acima são solicitadas ao usuário somente sua agência e conta. Após a digitação destes dados, o usuário deve clicar no botão conectar. O sistema, então, conecta-se à base de dados da instituição financeira. Quando a requisição chega na instituição é efetuada a validação de agência e conta recebidas. Após a validação, são buscadas no cadastro do usuário informações pessoais que serão utilizadas para geração das chaves públicas e privadas. As informações serão escolhidas randomicamente, dentre todas aquelas cadastradas, podendo ser telefone, filiação, naturalidade, hobbies, dentre outras.

Salienta-se que a aplicação do usuário só será responsável por criptografar a senha. Todo o processo de geração de chaves será feito na aplicação rodando no

servidor da instituição. Esta aplicação teria a seguinte aparência (não é necessária a tela apresentada abaixo, sendo mostrada apenas como ilustração):

Figura 11 – Aplicação Rodando no Servidor da Instituição – Versão Servidor

Na tela acima a aplicação já escolheu, aleatoriamente, duas informações que servirão de base para gerar os números primos do algoritmo RSA: nome da mãe e data de nascimento. Cada informação gerará um número primo diferente. Para o processo de escolha de cada número primo cada caracter da informação será transformado para seu código ASCII e será efetuada a soma de todos estes valores ASCII. Ao final da soma será encontrado um inteiro. O programa, então, percorrerá um loop, partindo do resultado, variando para baixo até encontrar um primo. Neste momento o loop é interrompido e a escolha é feita. Assim, considerando que a resposta para as informações acima sejam: “Maria” e “090579, sua representação em ASCII e seus primos correspondentes ficariam:

Maria = 77(M) + 97(a) + 116 (r) + 105 (i) + 97(a) = 492.

090579 = 48(0) + 57(9) + 48(0) + 53(5) + 55(7) + 57(9) = 318

Logo, o primeiro primo encontrado variando de 492 para menos (critério adotado, poderia ser para mais) é 487 e o segundo primo variando de 318 para menos é 317. Após a definição dos primos a aplicação já tem condições de escolher as chaves públicas e privadas necessárias para o restante do processo. Com a definição das chaves, apenas o par de chaves públicas é enviado à aplicação do usuário e com este par a chave é encriptada e enviada para a instituição. Após a recepção da senha encriptada, a aplicação rodando no servidor desencripta a senha com o par de chaves privadas e compara esta senha com a cadastrada na base de dados da instituição.

O processo descrito acima pode ser ilustrado da seguinte forma:

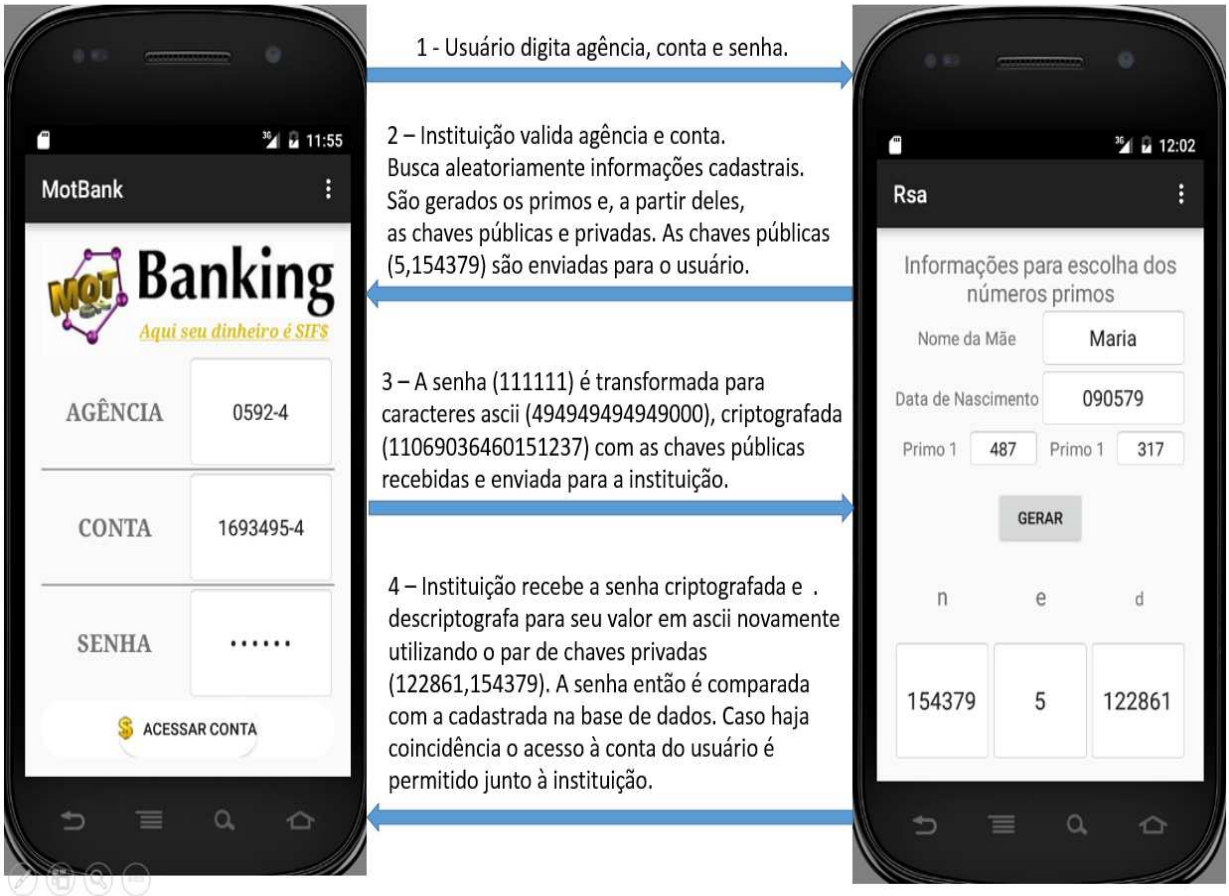


Figura 12 – Ilustração do Processo de Autenticação utilizando RSA

Acima foi demonstrado que apenas a chave criptografada é enviada pela rede. Ela foi gerada a partir das informações cadastrais escolhidas naquele momento: *nome da mãe e data de nascimento.* Como a escolha dos parâmetros que originam a criptografia da senha é aleatória, em um próximo acesso a escolha poderia ser o *nome do pai e telefone residencial*, gerando uma senha criptografada completamente diferente, conforme figura abaixo:

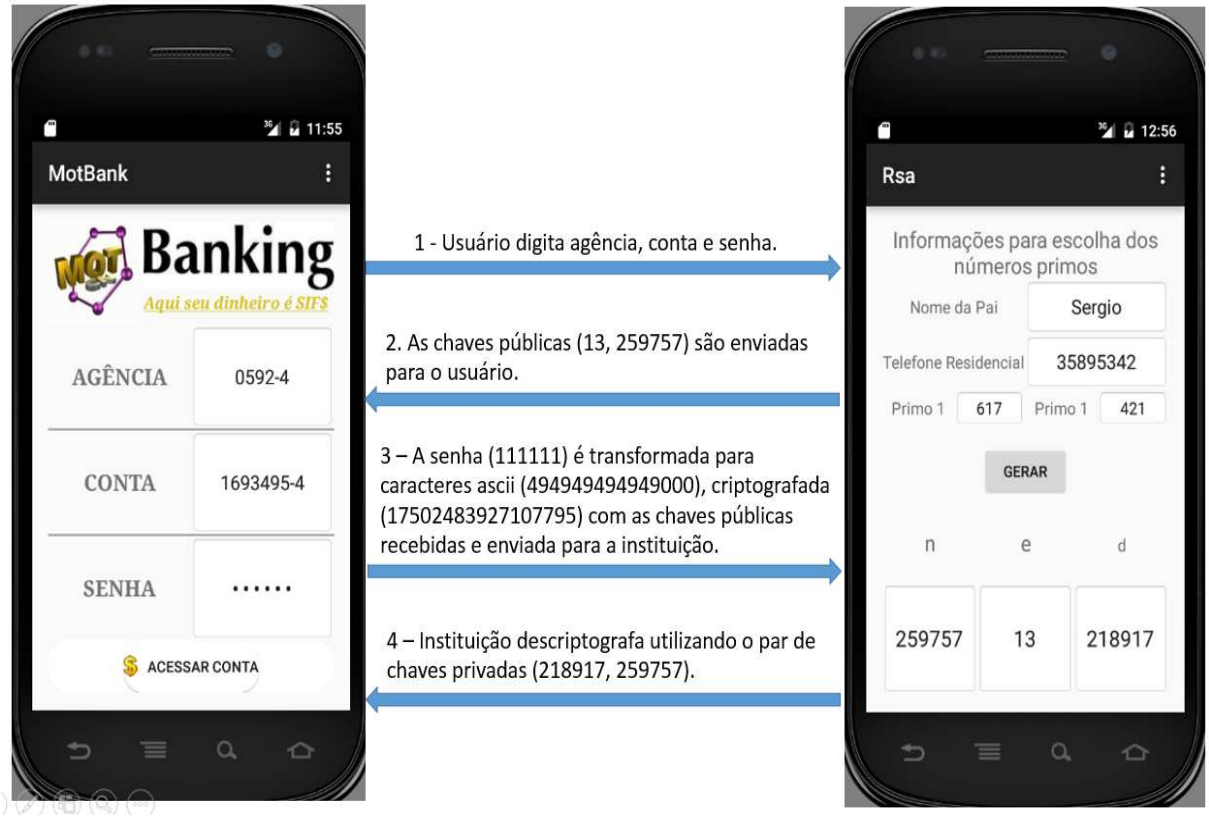


Figura 13 – Nova Criptografia Baseada em Informações Cadastrais Diferentes

6 **CONCLUSÃO**

O crescente aumento no volume de dados sigilos trafegando pela internet traz a necessidade do desenvolvimento de novas técnicas objetivando a proteção destes dados. Ilustrando este aumento, em 2014 as transações bancárias através da internet e mobile banking chegaram a 47%. Em 2009 eram apenas 31% (FREBABAN, 2014).

O objetivo deste estudo foi apresentar os principais conceitos e técnicas criptográficas e desenvolver um processo de autenticação que permite a criação de chaves criptográficas diferentes a cada tentativa de acesso de um cliente a sua conta em uma instituição financeira, simulando o método de acesso *OTP (*pois a cada conjunto de chaves diferentes geradas o resultado da codificação de uma mesma senha é diferente). No exemplo demonstrado no capítulo anterior foi evidenciada a praticidade do método. Caso a senha seja interceptada através de um ataque *man-in-the middle,* a mesma só pode ser descoberta através de um processo de “força bruta” ou através de um algoritmo de fatoração que tem um alto custo computacional para fatorar um número de módulo N. Outra vantagem, que pode vir a desestimular um hacker, é a simulação de que a senha varia a cada acesso. Assim, mesmo vindo a ser descoberta, não terá valia numa próxima autenticação já que ela seria alterada (OTP).

Foi desenvolvido um programa, na linguagem de programação Java, voltado para a plataforma móvel Android, onde foi implementado o processo proposto. O programa foi desenvolvido para adequar-se à arquitetura de dados de qualquer empresa, podendo ser aplicado para aplicações mobile quanto para aplicações web ou distribuídas. Para os testes realizados foram utilizadas chaves criptográficas de até 48 bits. O hardware rodando as aplicações possui a seguinte configuração:

1.2GHz de processador e memória RAM de 1GB. Para o tamanho de chaves mencionadas seu tempo de geração das chaves e criptografia das senhas foi da ordem de milissegundos.

Vários estudos mencionam que hoje para uma chave ser considerada segura ela deve ter no mínimo 1024 bits. Contudo deve-se levar em conta o esforço computacional para a geração destas chaves, além do custo de criptografar/descriptografar as informações com chaves de tal tamanho. Possivelmente pode tornar-se inviável trabalhar com chaves de comprimento elevado em aplicações mobile, por exemplo, onde há uma limitação do hardware que irá utilizar o processo. Deve-se sempre avaliar o custo de implementação x valor da informação que será protegida para definir a melhor estratégia de qual o tamanho de chaves desejado para o processo.

Este trabalho focou apenas na proposta e implementação do processo, sendo necessários estudos posteriores para avaliação da performance de utilização de chaves de tamanhos maiores e sua utilização em diversos tipos de hardware. Mas o que foi testado certamente garante proteção para a grande maioria dos casos, porque mesmo uma chave de 48 bits envolve conhecimento especializado para descoberta através do método de força bruta ou de fatoração além de simular o método OTP cuja geração aleatória de chaves e conseqüentemente da senha codificada dificulta a descoberta da mesma por eventuais hackers.

Essa geração aleatória dificulta a descoberta de senhas caso uma conexão esteja sendo interceptada constantemente pois não haverá um padrão na senha codificada sendo analisada (chaves diferentes geram resultados diferentes).

Outra proposta que pode ser desenvolvida em projetos posteriores, utilizando a mesma ideia deste estudo, é fazer todo o processo de geração das chaves e

codificação das senhas localmente na aplicação do usuário. Assim, após o processo de codificação, apenas a senha encriptada seria enviada pela rede e armazenada no banco de dados da instituição. No momento da autenticação a instituição enviaria a senha encriptada para a aplicação. A aplicação, por sua vez, solicitaria que o usuário digitasse a senha e também a encriptaria. Caso o resultado desta codificação fosse igual ao enviado pela instituição a autenticação seria efetuada. As vantagens deste método para o proposto no estudo são: a) as chaves públicas não são enviadas pela rede; b) pode ser feito um cadastro do usuário remotamente junto com o cadastro da senha, pois a mesma só é enviada à instituição na sua forma codificada. A desvantagem é o maior poder computacional exigido pelos hardwares que executam a aplicação do usuário já que todo o processo de geração e codificação é realizado localmente.

O presente estudo desenvolveu uma proposta de autenticação envolvendo a criptografia assimétrica RSA que é considerada a mais segura atualmente. Mas ela sozinha não garante a segurança que os usuários precisam em sua plenitude. Segurança é um processo e, como tal, envolve várias etapas para que seus objetivos sejam alcançados. Criptografia é apenas um deles. Outros que podem ser citados: colaboração do usuário, sistemas operacionais atualizados, utilização apenas de softwares conhecidos, digitação de informações confidenciais na web apenas em páginas certificadas por uma empresa de segurança digital, descarte adequado das informações, dentre outras. Aplicando todas estas medidas em conjunto o processo descrito neste estudo pode colaborar para o aumento da segurança digital em uma instituição que exige acesso restrito as suas informações confidenciais.

REFERÊNCIAS

[1] ALDEIA NUMABOA. 01 de outubro de 2005. Disponível em http://www.numaboa.com.br/criptografia/chaves/350-rsa?showall=1&limitstart=. Acesso em 17 de setembro de 2015.

[2] BARRETO, PAULO SERGIO L. M. **Criptografia Robusta e marcasd'agua frágeis: construção e análise de algoritmos para localizar alterações em imagens digitais.** Orientador: Prof. Dr. Hae Yong Kim. Escola Politécnica, Universidade de São Paulo, 2003. 72p.

[3] CHIRIGATI, F. S.; KIKUCHI, R. S. A.; GOMES, T. L. setembro/2009. Disponível em <http://www.gta.ufrj.br/grad/09_1/versao-final/stegano/index.html>. Acesso em 09 de junho de 2015.

[4] FEBRABAN. Abril/2014. Disponível em <https://febraban.org.br/Noticias1.asp?id_texto=2364>. Acesso em 16 de junho de 2016.

[5] FONG, DIANA. Fevereiro/2015. Disponível em <http://www.dw.de/poloneses-foram-os-primeiros-a-decifrar-c%C3%B3digo-enigma/a-18271543>. Acesso em 11 de junho de 2016.

[6] GUIMARÃES, CARLA ROCHA. **Criptografia para Segurança de Dados**. Orientador: Marcos Alberto Lopes da Silva, Msc. Centro Universitário do Triângulo - Unit, 2001. 20p.

[7] KHAN ACADEMY. [2---]. Disponível em<https://pt.khanacademy.org/computing/computer-science/cryptography/ciphers/a/ciphers-vs-codes>. Acesso em 12 de junho de 2015.

[8] KRISCHER, THAIS CRISTINE. **Um estudo da máquina enigma**. Orientador: Raul Fernando Weber. Trabalho de Graduação (Bacharelado em Ciências da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, 2013. 19p.

[9] MORAES, ROSANA FRANÇA DE. **Construção de um ambiente web com ferramentas para estudo de algoritmos de criptografia através do Matlab.**Orientador: Luis Felipe M. de Moraes, Ph.D. Escola de Engenharia, Universidade Federal do Rio de Janeiro, 2004. 62p.

[10] PIGATTO, DANIEL FERNANDO. **Segurança em sistemas embarcados críticos – utilização de criptografia para comunicação segura**. Orientadora: Profa. Dra. Kalinka Regina Lucas Jaquie Castelo Branco. Universidade de São Paulo – São Carlos, 2012. 18p.

[11] PROF. CARDY. [2---]. Disponível em http://www.profcardy.com/cardicas/euclides.php. Acesso em 04 de novembro de 2015.

[12] ROCHA, ANDERSON DE REZENDE. **Camaleão: Um software para segurança digital utilizando esteganografia.** Orientador: Heitor Augustus Xavier Costa. Universidade Federal de Lavras, 2003. 79p.

[13] SEGURANÇA DA INFORMAÇÃO. [2---]. Disponível em http://seguranca-da-informacao.info/criptografia.html. Acesso em 08 de dezembro de 2015.

[14] TRINTA, F. A .M., MACÊDO, R. C. Um estudo sobre criptografia e assinatura digital. [1998]. Disponível em http://www.di.ufpe.br/~flash/ais98/cripto/criptografia.htm. Departamento de Informática, Universidade Federal de Pernambuco. Acesso em 08 de dezembro de 2015.

[15] VIANNA, Ricardo. Maio/2011. Disponível em <http://prof-ricardovianna.blogspot.com.br/2011/05/criptografia-parte-i-historia-da.html>. Acesso em 10 de junho de 2015.

GLOSSÁRIO

Bit de paridade: último bit de uma cadeia de caracter. Caso a quantidade de bits“1” da cadeia seja par o bit de paridade tem o valor “1”, caso a quantidade seja ímpar o bit assume o valor “0”.

***Backdoor*:** é um recurso instalado na máquina do usuário que permite o controle desta máquina por um atacante localizado remotamente.

Código ASC II: Código que transforma cada caracter conhecido pela linguagem humana em seu equivalente numérico que é reconhecido pelo linguagem de máquina.

OTP: Sistema de autenticação por senha, onde cada senha é utilizada uma única vez para acesso. Após o seu uso ela é descartada e gerada uma nova.

***Man-in-the-middle*:** tipo de ataque virtual onde um potencial intruso insere-se em uma comunicação entre remetente e destinatário, analisando todos os pacotes desta conexão.

APÊNDICE A – Programação do algoritmo DES

a) A partir de uma chave K de 56 bits, criar 16 subchaves de 48 bits (após o cálculo de Cn, Dn):

```
Sub CalculaK()

For i = 2 To 17

  Plan1.Cells(i, 6) = Mid(Plan1.Cells(i, 7), 14, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 17, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 11, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 24, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 1, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 5, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 3, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 28, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 15, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 6, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 21, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 10, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 23, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 19, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 12, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 4, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 26, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 8, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 16, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 7, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 27, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 20, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 13, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 2, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 41, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 52, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 31, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 37, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 47, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 55, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 30, 1)
  Plan1.Cells(i, 6) = Plan1.Cells(i, 6) + Mid(Plan1.Cells(i, 7), 40, 1)

Next i

End Sub
```

b) Codificar cada bloco de 64 bits de dados:

```
Public MatrizS1(0 To 3, 0 To 15) As Byte
Public MatrizS2(0 To 3, 0 To 15) As Byte
Public MatrizS3(0 To 3, 0 To 15) As Byte
Public MatrizS4(0 To 3, 0 To 15) As Byte
Public MatrizS5(0 To 3, 0 To 15) As Byte
Public MatrizS6(0 To 3, 0 To 15) As Byte
Public MatrizS7(0 To 3, 0 To 15) As Byte
Public MatrizS8(0 To 3, 0 To 15) As Byte

Sub Des()
Dim Expansao As String
Dim VarXor As String
Dim S1 As String
Dim S2 As String
Dim S3 As String
Dim S4 As String
Dim S5 As String
Dim S6 As String
Dim S7 As String
Dim S8 As String
Dim resultadoF As String
Dim fPermutado As String
Dim concatenaRL As String
Dim resultadoFinal As String

  'Procedimento para preencher as matrizes das funções S1 a S8
  Call PMatrizS1
  Call PMatrizS2
  Call PMatrizS3
  Call PMatrizS4
  Call PMatrizS5
  Call PMatrizS6
  Call PMatrizS7
  Call PMatrizS8

  For i = 2 To 16
    'Cálculo do valor de Li = Ri-1
    Plan2.Cells(i, 2) = Plan2.Cells(i - 1, 4)

    'Cálculo da expansão de R0 para 48 bits
    Expansao = ExpR0(Plan2.Cells(i - 1, 4))

    'Calculo do XOR entre Rexpandido e K = 48 bits
    VarXor = FXor(Expansao, Plan2.Cells(i, 6))
```

```
      ' Calculo das Funções S1 até S8 retornando com 4 bits
      S1 = FS1(Mid(VarXor, 1, 6))
      S2 = FS2(Mid(VarXor, 7, 6))
      S3 = FS3(Mid(VarXor, 13, 6))
      S4 = FS4(Mid(VarXor, 19, 6))
      S5 = FS5(Mid(VarXor, 25, 6))
      S6 = FS6(Mid(VarXor, 31, 6))
      S7 = FS7(Mid(VarXor, 37, 6))
      S8 = FS8(Mid(VarXor, 43, 6))

      resultadoF = S1 + S2 + S3 + S4 + S5 + S6 + S7 + S8
      fPermutado = PermutaF(resultadoF)

      'Calcula Ri
      Plan2.Cells(i, 4) = Xor32(Plan2.Cells(i - 1, 2), fPermutado)

      'Concatena R e L na última iteração
      If i = 16 Then
        concatenaRL = Plan2.Cells(i, 4) + Plan2.Cells(i, 2)

        'Permuta a concatenação de R e L
       resultadoFinal = PermutaRL(concatenaRL)
       Plan2.Cells(23, 6) = resultadoFinal

      End If

    Next i

End Sub

**********************************************************************
Function Xor32(L1 As String, fPermutado As String)
  Xor32 = ""

For i = 1 To 32
  If Mid(L1, i, 1) = Mid(fPermutado, i, 1) Then
    Xor32 = Xor32 + "0"
  Else
    Xor32 = Xor32 + "1"
  End If
Next i
End Function

**********************************************************************

Function PermutaRL(cadeia As String)
  PermutaRL = Mid(cadeia, 40, 1)
```

```
PermutaRL = PermutaRL + Mid(cadeia, 8, 1)
PermutaRL = PermutaRL + Mid(cadeia, 48, 1)
PermutaRL = PermutaRL + Mid(cadeia, 16, 1)
PermutaRL = PermutaRL + Mid(cadeia, 56, 1)
PermutaRL = PermutaRL + Mid(cadeia, 24, 1)
PermutaRL = PermutaRL + Mid(cadeia, 64, 1)
PermutaRL = PermutaRL + Mid(cadeia, 32, 1)
PermutaRL = PermutaRL + Mid(cadeia, 39, 1)
PermutaRL = PermutaRL + Mid(cadeia, 7, 1)
PermutaRL = PermutaRL + Mid(cadeia, 47, 1)
PermutaRL = PermutaRL + Mid(cadeia, 15, 1)
PermutaRL = PermutaRL + Mid(cadeia, 55, 1)
PermutaRL = PermutaRL + Mid(cadeia, 23, 1)
PermutaRL = PermutaRL + Mid(cadeia, 63, 1)
PermutaRL = PermutaRL + Mid(cadeia, 31, 1)
PermutaRL = PermutaRL + Mid(cadeia, 38, 1)
PermutaRL = PermutaRL + Mid(cadeia, 6, 1)
PermutaRL = PermutaRL + Mid(cadeia, 46, 1)
PermutaRL = PermutaRL + Mid(cadeia, 14, 1)
PermutaRL = PermutaRL + Mid(cadeia, 54, 1)
PermutaRL = PermutaRL + Mid(cadeia, 22, 1)
PermutaRL = PermutaRL + Mid(cadeia, 62, 1)
PermutaRL = PermutaRL + Mid(cadeia, 30, 1)
PermutaRL = PermutaRL + Mid(cadeia, 37, 1)
PermutaRL = PermutaRL + Mid(cadeia, 5, 1)
PermutaRL = PermutaRL + Mid(cadeia, 45, 1)
PermutaRL = PermutaRL + Mid(cadeia, 13, 1)
PermutaRL = PermutaRL + Mid(cadeia, 53, 1)
PermutaRL = PermutaRL + Mid(cadeia, 21, 1)
PermutaRL = PermutaRL + Mid(cadeia, 61, 1)
PermutaRL = PermutaRL + Mid(cadeia, 29, 1)
PermutaRL = PermutaRL + Mid(cadeia, 36, 1)
PermutaRL = PermutaRL + Mid(cadeia, 4, 1)
PermutaRL = PermutaRL + Mid(cadeia, 44, 1)
PermutaRL = PermutaRL + Mid(cadeia, 12, 1)
PermutaRL = PermutaRL + Mid(cadeia, 52, 1)
PermutaRL = PermutaRL + Mid(cadeia, 20, 1)
PermutaRL = PermutaRL + Mid(cadeia, 60, 1)
PermutaRL = PermutaRL + Mid(cadeia, 28, 1)
PermutaRL = PermutaRL + Mid(cadeia, 35, 1)
PermutaRL = PermutaRL + Mid(cadeia, 3, 1)
PermutaRL = PermutaRL + Mid(cadeia, 43, 1)
PermutaRL = PermutaRL + Mid(cadeia, 11, 1)
PermutaRL = PermutaRL + Mid(cadeia, 51, 1)
PermutaRL = PermutaRL + Mid(cadeia, 19, 1)
PermutaRL = PermutaRL + Mid(cadeia, 59, 1)
PermutaRL = PermutaRL + Mid(cadeia, 27, 1)
```

```
  PermutaRL = PermutaRL + Mid(cadeia, 34, 1)
  PermutaRL = PermutaRL + Mid(cadeia, 2, 1)
  PermutaRL = PermutaRL + Mid(cadeia, 42, 1)
  PermutaRL = PermutaRL + Mid(cadeia, 10, 1)
  PermutaRL = PermutaRL + Mid(cadeia, 50, 1)
  PermutaRL = PermutaRL + Mid(cadeia, 18, 1)
  PermutaRL = PermutaRL + Mid(cadeia, 58, 1)
  PermutaRL = PermutaRL + Mid(cadeia, 26, 1)
  PermutaRL = PermutaRL + Mid(cadeia, 33, 1)
  PermutaRL = PermutaRL + Mid(cadeia, 1, 1)
  PermutaRL = PermutaRL + Mid(cadeia, 41, 1)
  PermutaRL = PermutaRL + Mid(cadeia, 9, 1)
  PermutaRL = PermutaRL + Mid(cadeia, 49, 1)
  PermutaRL = PermutaRL + Mid(cadeia, 17, 1)
  PermutaRL = PermutaRL + Mid(cadeia, 57, 1)
  PermutaRL = PermutaRL + Mid(cadeia, 25, 1)
End Function

**********************************************************************

Function PermutaF(f As String)
  PermutaF = Mid(f, 16, 1)
  PermutaF = PermutaF + Mid(f, 7, 1)
  PermutaF = PermutaF + Mid(f, 20, 1)
  PermutaF = PermutaF + Mid(f, 1, 1)
  PermutaF = PermutaF + Mid(f, 29, 1)
  PermutaF = PermutaF + Mid(f, 12, 1)
  PermutaF = PermutaF + Mid(f, 28, 1)
  PermutaF = PermutaF + Mid(f, 17, 1)
  PermutaF = PermutaF + Mid(f, 1, 1)
  PermutaF = PermutaF + Mid(f, 15, 1)
  PermutaF = PermutaF + Mid(f, 23, 1)
  PermutaF = PermutaF + Mid(f, 26, 1)
  PermutaF = PermutaF + Mid(f, 5, 1)
  PermutaF = PermutaF + Mid(f, 18, 1)
  PermutaF = PermutaF + Mid(f, 31, 1)
  PermutaF = PermutaF + Mid(f, 10, 1)
  PermutaF = PermutaF + Mid(f, 2, 1)
  PermutaF = PermutaF + Mid(f, 8, 1)
  PermutaF = PermutaF + Mid(f, 24, 1)
  PermutaF = PermutaF + Mid(f, 14, 1)
  PermutaF = PermutaF + Mid(f, 32, 1)
  PermutaF = PermutaF + Mid(f, 27, 1)
  PermutaF = PermutaF + Mid(f, 3, 1)
  PermutaF = PermutaF + Mid(f, 9, 1)
  PermutaF = PermutaF + Mid(f, 19, 1)
  PermutaF = PermutaF + Mid(f, 13, 1)
```

```
    PermutaF = PermutaF + Mid(f, 30, 1)
    PermutaF = PermutaF + Mid(f, 6, 1)
    PermutaF = PermutaF + Mid(f, 22, 1)
    PermutaF = PermutaF + Mid(f, 11, 1)
    PermutaF = PermutaF + Mid(f, 4, 1)
    PermutaF = PermutaF + Mid(f, 25, 1)
End Function

**************************************************************************

Sub PMatrizS1()
    MatrizS1(0, 0) = 14
    MatrizS1(0, 1) = 4
    MatrizS1(0, 2) = 13
    MatrizS1(0, 3) = 1
    MatrizS1(0, 4) = 2
    MatrizS1(0, 5) = 15
    MatrizS1(0, 6) = 11
    MatrizS1(0, 7) = 8
    MatrizS1(0, 8) = 3
    MatrizS1(0, 9) = 10
    MatrizS1(0, 10) = 6
    MatrizS1(0, 11) = 12
    MatrizS1(0, 12) = 5
    MatrizS1(0, 13) = 9
    MatrizS1(0, 14) = 0
    MatrizS1(0, 15) = 7
    MatrizS1(1, 0) = 0
    MatrizS1(1, 1) = 15
    MatrizS1(1, 2) = 7
    MatrizS1(1, 3) = 4
    MatrizS1(1, 4) = 14
    MatrizS1(1, 5) = 12
    MatrizS1(1, 6) = 13
    MatrizS1(1, 7) = 1
    MatrizS1(1, 8) = 10
    MatrizS1(1, 9) = 6
    MatrizS1(1, 10) = 12
    MatrizS1(1, 11) = 11
    MatrizS1(1, 12) = 9
    MatrizS1(1, 13) = 5
    MatrizS1(1, 14) = 3
    MatrizS1(1, 15) = 8
    MatrizS1(2, 0) = 4
    MatrizS1(2, 1) = 1
    MatrizS1(2, 2) = 14
    MatrizS1(2, 3) = 8
```

```
    MatrizS1(2, 4) = 13
    MatrizS1(2, 5) = 6
    MatrizS1(2, 6) = 2
    MatrizS1(2, 7) = 11
    MatrizS1(2, 8) = 15
    MatrizS1(2, 9) = 12
    MatrizS1(2, 10) = 9
    MatrizS1(2, 11) = 7
    MatrizS1(2, 12) = 3
    MatrizS1(2, 13) = 10
    MatrizS1(2, 14) = 5
    MatrizS1(2, 15) = 0
    MatrizS1(3, 0) = 15
    MatrizS1(3, 1) = 12
    MatrizS1(3, 2) = 8
    MatrizS1(3, 3) = 2
    MatrizS1(3, 4) = 4
    MatrizS1(3, 5) = 9
    MatrizS1(3, 6) = 1
    MatrizS1(3, 7) = 7
    MatrizS1(3, 8) = 5
    MatrizS1(3, 9) = 11
    MatrizS1(3, 10) = 3
    MatrizS1(3, 11) = 14
    MatrizS1(3, 12) = 10
    MatrizS1(3, 13) = 6
    MatrizS1(3, 14) = 6
    MatrizS1(3, 15) = 13

End Sub

Sub PMatrizS2()
    MatrizS2(0, 0) = 15
    MatrizS2(0, 1) = 1
    MatrizS2(0, 2) = 8
    MatrizS2(0, 3) = 14
    MatrizS2(0, 4) = 6
    MatrizS2(0, 5) = 11
    MatrizS2(0, 6) = 3
    MatrizS2(0, 7) = 4
    MatrizS2(0, 8) = 9
    MatrizS2(0, 9) = 7
    MatrizS2(0, 10) = 2
    MatrizS2(0, 11) = 13
    MatrizS2(0, 12) = 12
    MatrizS2(0, 13) = 0
    MatrizS2(0, 14) = 5
```

MatrizS2(0, 15) = 10
MatrizS2(1, 0) = 3
MatrizS2(1, 1) = 13
MatrizS2(1, 2) = 4
MatrizS2(1, 3) = 7
MatrizS2(1, 4) = 15
MatrizS2(1, 5) = 2
MatrizS2(1, 6) = 8
MatrizS2(1, 7) = 14
MatrizS2(1, 8) = 12
MatrizS2(1, 9) = 0
MatrizS2(1, 10) = 1
MatrizS2(1, 11) = 10
MatrizS2(1, 12) = 6
MatrizS2(1, 13) = 9
MatrizS2(1, 14) = 11
MatrizS2(1, 15) = 5
MatrizS2(2, 0) = 0
MatrizS2(2, 1) = 14
MatrizS2(2, 2) = 7
MatrizS2(2, 3) = 11
MatrizS2(2, 4) = 10
MatrizS2(2, 5) = 4
MatrizS2(2, 6) = 13
MatrizS2(2, 7) = 1
MatrizS2(2, 8) = 5
MatrizS2(2, 9) = 8
MatrizS2(2, 10) = 12
MatrizS2(2, 11) = 6
MatrizS2(2, 12) = 9
MatrizS2(2, 13) = 3
MatrizS2(2, 14) = 2
MatrizS2(2, 15) = 15
MatrizS2(3, 0) = 13
MatrizS2(3, 1) = 8
MatrizS2(3, 2) = 10
MatrizS2(3, 3) = 1
MatrizS2(3, 4) = 3
MatrizS2(3, 5) = 15
MatrizS2(3, 6) = 4
MatrizS2(3, 7) = 2
MatrizS2(3, 8) = 11
MatrizS2(3, 9) = 6
MatrizS2(3, 10) = 7
MatrizS2(3, 11) = 12
MatrizS2(3, 12) = 0
MatrizS2(3, 13) = 5

```
    MatrizS2(3, 14) = 14
    MatrizS2(3, 15) = 9

End Sub

Sub PMatrizS3()
    MatrizS3(0, 0) = 10
    MatrizS3(0, 1) = 0
    MatrizS3(0, 2) = 9
    MatrizS3(0, 3) = 14
    MatrizS3(0, 4) = 6
    MatrizS3(0, 5) = 3
    MatrizS3(0, 6) = 15
    MatrizS3(0, 7) = 5
    MatrizS3(0, 8) = 1
    MatrizS3(0, 9) = 13
    MatrizS3(0, 10) = 12
    MatrizS3(0, 11) = 7
    MatrizS3(0, 12) = 11
    MatrizS3(0, 13) = 4
    MatrizS3(0, 14) = 2
    MatrizS3(0, 15) = 8
    MatrizS3(1, 0) = 17
    MatrizS3(1, 1) = 7
    MatrizS3(1, 2) = 0
    MatrizS3(1, 3) = 9
    MatrizS3(1, 4) = 3
    MatrizS3(1, 5) = 4
    MatrizS3(1, 6) = 6
    MatrizS3(1, 7) = 10
    MatrizS3(1, 8) = 2
    MatrizS3(1, 9) = 8
    MatrizS3(1, 10) = 5
    MatrizS3(1, 11) = 14
    MatrizS3(1, 12) = 12
    MatrizS3(1, 13) = 11
    MatrizS3(1, 14) = 15
    MatrizS3(1, 15) = 1
    MatrizS3(2, 0) = 13
    MatrizS3(2, 1) = 6
    MatrizS3(2, 2) = 4
    MatrizS3(2, 3) = 9
    MatrizS3(2, 4) = 8
    MatrizS3(2, 5) = 15
    MatrizS3(2, 6) = 3
    MatrizS3(2, 7) = 0
    MatrizS3(2, 8) = 11
```

```
    MatrizS3(2, 9) = 1
    MatrizS3(2, 10) = 2
    MatrizS3(2, 11) = 12
    MatrizS3(2, 12) = 5
    MatrizS3(2, 13) = 10
    MatrizS3(2, 14) = 14
    MatrizS3(2, 15) = 7
    MatrizS3(3, 0) = 1
    MatrizS3(3, 1) = 10
    MatrizS3(3, 2) = 13
    MatrizS3(3, 3) = 0
    MatrizS3(3, 4) = 6
    MatrizS3(3, 5) = 9
    MatrizS3(3, 6) = 8
    MatrizS3(3, 7) = 7
    MatrizS3(3, 8) = 4
    MatrizS3(3, 9) = 15
    MatrizS3(3, 10) = 14
    MatrizS3(3, 11) = 3
    MatrizS3(3, 12) = 11
    MatrizS3(3, 13) = 5
    MatrizS3(3, 14) = 2
    MatrizS3(3, 15) = 12

End Sub

Sub PMatrizS4()
    MatrizS4(0, 0) = 7
    MatrizS4(0, 1) = 13
    MatrizS4(0, 2) = 14
    MatrizS4(0, 3) = 3
    MatrizS4(0, 4) = 0
    MatrizS4(0, 5) = 6
    MatrizS4(0, 6) = 9
    MatrizS4(0, 7) = 10
    MatrizS4(0, 8) = 1
    MatrizS4(0, 9) = 2
    MatrizS4(0, 10) = 8
    MatrizS4(0, 11) = 5
    MatrizS4(0, 12) = 11
    MatrizS4(0, 13) = 12
    MatrizS4(0, 14) = 4
    MatrizS4(0, 15) = 15
    MatrizS4(1, 0) = 13
    MatrizS4(1, 1) = 8
    MatrizS4(1, 2) = 11
    MatrizS4(1, 3) = 5
```

```
    MatrizS4(1, 4) = 6
    MatrizS4(1, 5) = 15
    MatrizS4(1, 6) = 0
    MatrizS4(1, 7) = 3
    MatrizS4(1, 8) = 4
    MatrizS4(1, 9) = 7
    MatrizS4(1, 10) = 2
    MatrizS4(1, 11) = 12
    MatrizS4(1, 12) = 1
    MatrizS4(1, 13) = 10
    MatrizS4(1, 14) = 14
    MatrizS4(1, 15) = 9
    MatrizS4(2, 0) = 10
    MatrizS4(2, 1) = 6
    MatrizS4(2, 2) = 9
    MatrizS4(2, 3) = 0
    MatrizS4(2, 4) = 12
    MatrizS4(2, 5) = 11
    MatrizS4(2, 6) = 7
    MatrizS4(2, 7) = 13
    MatrizS4(2, 8) = 15
    MatrizS4(2, 9) = 1
    MatrizS4(2, 10) = 3
    MatrizS4(2, 11) = 14
    MatrizS4(2, 12) = 5
    MatrizS4(2, 13) = 2
    MatrizS4(2, 14) = 8
    MatrizS4(2, 15) = 4
    MatrizS4(3, 0) = 3
    MatrizS4(3, 1) = 15
    MatrizS4(3, 2) = 0
    MatrizS4(3, 3) = 6
    MatrizS4(3, 4) = 10
    MatrizS4(3, 5) = 1
    MatrizS4(3, 6) = 13
    MatrizS4(3, 7) = 8
    MatrizS4(3, 8) = 9
    MatrizS4(3, 9) = 4
    MatrizS4(3, 10) = 5
    MatrizS4(3, 11) = 11
    MatrizS4(3, 12) = 12
    MatrizS4(3, 13) = 7
    MatrizS4(3, 14) = 2
    MatrizS4(3, 15) = 14

End Sub
```

```
Sub PMatrizS5()
   MatrizS5(0, 0) = 2
   MatrizS5(0, 1) = 12
   MatrizS5(0, 2) = 4
   MatrizS5(0, 3) = 1
   MatrizS5(0, 4) = 7
   MatrizS5(0, 5) = 10
   MatrizS5(0, 6) = 11
   MatrizS5(0, 7) = 6
   MatrizS5(0, 8) = 8
   MatrizS5(0, 9) = 5
   MatrizS5(0, 10) = 3
   MatrizS5(0, 11) = 15
   MatrizS5(0, 12) = 13
   MatrizS5(0, 13) = 0
   MatrizS5(0, 14) = 14
   MatrizS5(0, 15) = 9
   MatrizS5(1, 0) = 14
   MatrizS5(1, 1) = 11
   MatrizS5(1, 2) = 2
   MatrizS5(1, 3) = 12
   MatrizS5(1, 4) = 4
   MatrizS5(1, 5) = 7
   MatrizS5(1, 6) = 13
   MatrizS5(1, 7) = 1
   MatrizS5(1, 8) = 5
   MatrizS5(1, 9) = 0
   MatrizS5(1, 10) = 15
   MatrizS5(1, 11) = 10
   MatrizS5(1, 12) = 3
   MatrizS5(1, 13) = 9
   MatrizS5(1, 14) = 8
   MatrizS5(1, 15) = 6
   MatrizS5(2, 0) = 4
   MatrizS5(2, 1) = 2
   MatrizS5(2, 2) = 1
   MatrizS5(2, 3) = 11
   MatrizS5(2, 4) = 10
   MatrizS5(2, 5) = 13
   MatrizS5(2, 6) = 7
   MatrizS5(2, 7) = 8
   MatrizS5(2, 8) = 15
   MatrizS5(2, 9) = 9
   MatrizS5(2, 10) = 12
   MatrizS5(2, 11) = 5
   MatrizS5(2, 12) = 6
   MatrizS5(2, 13) = 0
```

```
    MatrizS5(2, 14) = 3
    MatrizS5(2, 15) = 14
    MatrizS5(3, 0) = 11
    MatrizS5(3, 1) = 8
    MatrizS5(3, 2) = 12
    MatrizS5(3, 3) = 7
    MatrizS5(3, 4) = 1
    MatrizS5(3, 5) = 14
    MatrizS5(3, 6) = 2
    MatrizS5(3, 7) = 13
    MatrizS5(3, 8) = 6
    MatrizS5(3, 9) = 15
    MatrizS5(3, 10) = 0
    MatrizS5(3, 11) = 9
    MatrizS5(3, 12) = 10
    MatrizS5(3, 13) = 4
    MatrizS5(3, 14) = 5
    MatrizS5(3, 15) = 3

End Sub

Sub PMatrizS6()
    MatrizS6(0, 0) = 12
    MatrizS6(0, 1) = 1
    MatrizS6(0, 2) = 10
    MatrizS6(0, 3) = 15
    MatrizS6(0, 4) = 9
    MatrizS6(0, 5) = 2
    MatrizS6(0, 6) = 6
    MatrizS6(0, 7) = 8
    MatrizS6(0, 8) = 0
    MatrizS6(0, 9) = 13
    MatrizS6(0, 10) = 3
    MatrizS6(0, 11) = 4
    MatrizS6(0, 12) = 14
    MatrizS6(0, 13) = 7
    MatrizS6(0, 14) = 5
    MatrizS6(0, 15) = 11
    MatrizS6(1, 0) = 10
    MatrizS6(1, 1) = 15
    MatrizS6(1, 2) = 4
    MatrizS6(1, 3) = 2
    MatrizS6(1, 4) = 7
    MatrizS6(1, 5) = 12
    MatrizS6(1, 6) = 9
    MatrizS6(1, 7) = 5
    MatrizS6(1, 8) = 6
```

```
    MatrizS6(1, 9) = 1
    MatrizS6(1, 10) = 13
    MatrizS6(1, 11) = 14
    MatrizS6(1, 12) = 0
    MatrizS6(1, 13) = 11
    MatrizS6(1, 14) = 3
    MatrizS6(1, 15) = 8
    MatrizS6(2, 0) = 9
    MatrizS6(2, 1) = 14
    MatrizS6(2, 2) = 15
    MatrizS6(2, 3) = 5
    MatrizS6(2, 4) = 2
    MatrizS6(2, 5) = 8
    MatrizS6(2, 6) = 12
    MatrizS6(2, 7) = 3
    MatrizS6(2, 8) = 7
    MatrizS6(2, 9) = 0
    MatrizS6(2, 10) = 4
    MatrizS6(2, 11) = 10
    MatrizS6(2, 12) = 1
    MatrizS6(2, 13) = 13
    MatrizS6(2, 14) = 11
    MatrizS6(2, 15) = 6
    MatrizS6(3, 0) = 4
    MatrizS6(3, 1) = 3
    MatrizS6(3, 2) = 2
    MatrizS6(3, 3) = 12
    MatrizS6(3, 4) = 9
    MatrizS6(3, 5) = 5
    MatrizS6(3, 6) = 15
    MatrizS6(3, 7) = 10
    MatrizS6(3, 8) = 11
    MatrizS6(3, 9) = 14
    MatrizS6(3, 10) = 1
    MatrizS6(3, 11) = 7
    MatrizS6(3, 12) = 6
    MatrizS6(3, 13) = 0
    MatrizS6(3, 14) = 8
    MatrizS6(3, 15) = 13

End Sub
```

```
Sub PMatrizS7()
  MatrizS7(0, 0) = 4
  MatrizS7(0, 1) = 11
  MatrizS7(0, 2) = 2
  MatrizS7(0, 3) = 14
  MatrizS7(0, 4) = 15
  MatrizS7(0, 5) = 0
  MatrizS7(0, 6) = 8
  MatrizS7(0, 7) = 13
  MatrizS7(0, 8) = 3
  MatrizS7(0, 9) = 12
  MatrizS7(0, 10) = 9
  MatrizS7(0, 11) = 7
  MatrizS7(0, 12) = 5
  MatrizS7(0, 13) = 10
  MatrizS7(0, 14) = 6
  MatrizS7(0, 15) = 1
  MatrizS7(1, 0) = 13
  MatrizS7(1, 1) = 0
  MatrizS7(1, 2) = 11
  MatrizS7(1, 3) = 7
  MatrizS7(1, 4) = 4
  MatrizS7(1, 5) = 9
  MatrizS7(1, 6) = 1
  MatrizS7(1, 7) = 10
  MatrizS7(1, 8) = 14
  MatrizS7(1, 9) = 3
  MatrizS7(1, 10) = 5
  MatrizS7(1, 11) = 12
  MatrizS7(1, 12) = 2
  MatrizS7(1, 13) = 15
  MatrizS7(1, 14) = 8
  MatrizS7(1, 15) = 6
  MatrizS7(2, 0) = 1
  MatrizS7(2, 1) = 4
  MatrizS7(2, 2) = 11
  MatrizS7(2, 3) = 13
  MatrizS7(2, 4) = 12
  MatrizS7(2, 5) = 3
  MatrizS7(2, 6) = 7
  MatrizS7(2, 7) = 14
  MatrizS7(2, 8) = 10
  MatrizS7(2, 9) = 15
  MatrizS7(2, 10) = 6
  MatrizS7(2, 11) = 8
  MatrizS7(2, 12) = 0
  MatrizS7(2, 13) = 5
```

```
    MatrizS7(2, 14) = 9
    MatrizS7(2, 15) = 2
    MatrizS7(3, 0) = 6
    MatrizS7(3, 1) = 11
    MatrizS7(3, 2) = 13
    MatrizS7(3, 3) = 8
    MatrizS7(3, 4) = 1
    MatrizS7(3, 5) = 4
    MatrizS7(3, 6) = 10
    MatrizS7(3, 7) = 7
    MatrizS7(3, 8) = 9
    MatrizS7(3, 9) = 5
    MatrizS7(3, 10) = 0
    MatrizS7(3, 11) = 15
    MatrizS7(3, 12) = 14
    MatrizS7(3, 13) = 2
    MatrizS7(3, 14) = 3
    MatrizS7(3, 15) = 12

End Sub

Sub PMatrizS8()
    MatrizS8(0, 0) = 13
    MatrizS8(0, 1) = 12
    MatrizS8(0, 2) = 8
    MatrizS8(0, 3) = 4
    MatrizS8(0, 4) = 6
    MatrizS8(0, 5) = 15
    MatrizS8(0, 6) = 11
    MatrizS8(0, 7) = 1
    MatrizS8(0, 8) = 10
    MatrizS8(0, 9) = 9
    MatrizS8(0, 10) = 3
    MatrizS8(0, 11) = 14
    MatrizS8(0, 12) = 5
    MatrizS8(0, 13) = 0
    MatrizS8(0, 14) = 12
    MatrizS8(0, 15) = 7
    MatrizS8(1, 0) = 1
    MatrizS8(1, 1) = 15
    MatrizS8(1, 2) = 13
    MatrizS8(1, 3) = 8
    MatrizS8(1, 4) = 10
    MatrizS8(1, 5) = 3
    MatrizS8(1, 6) = 7
    MatrizS8(1, 7) = 4
    MatrizS8(1, 8) = 12
```

```
  MatrizS8(1, 9) = 5
  MatrizS8(1, 10) = 6
  MatrizS8(1, 11) = 11
  MatrizS8(1, 12) = 0
  MatrizS8(1, 13) = 14
  MatrizS8(1, 14) = 9
  MatrizS8(1, 15) = 2
  MatrizS8(2, 0) = 7
  MatrizS8(2, 1) = 11
  MatrizS8(2, 2) = 4
  MatrizS8(2, 3) = 1
  MatrizS8(2, 4) = 9
  MatrizS8(2, 5) = 12
  MatrizS8(2, 6) = 14
  MatrizS8(2, 7) = 2
  MatrizS8(2, 8) = 0
  MatrizS8(2, 9) = 6
  MatrizS8(2, 10) = 10
  MatrizS8(2, 11) = 13
  MatrizS8(2, 12) = 15
  MatrizS8(2, 13) = 3
  MatrizS8(2, 14) = 5
  MatrizS8(2, 15) = 8
  MatrizS8(3, 0) = 2
  MatrizS8(3, 1) = 1
  MatrizS8(3, 2) = 14
  MatrizS8(3, 3) = 7
  MatrizS8(3, 4) = 4
  MatrizS8(3, 5) = 10
  MatrizS8(3, 6) = 8
  MatrizS8(3, 7) = 13
  MatrizS8(3, 8) = 15
  MatrizS8(3, 9) = 12
  MatrizS8(3, 10) = 9
  MatrizS8(3, 11) = 0
  MatrizS8(3, 12) = 3
  MatrizS8(3, 13) = 5
  MatrizS8(3, 14) = 6
  MatrizS8(3, 15) = 11

End Sub

**********************************************************************
```

```
'Função para calcular XOR entre K e R
Function FXor(r As String, K As String) As String
FXor = ""

For i = 1 To 48
  If Mid(r, i, 1) = Mid(K, i, 1) Then
    FXor = FXor + "0"
  Else
    FXor = FXor + "1"
  End If
Next i

End Function

**********************************************************************

'Função para expandir R de 32 para 48 bits
Function ExpR0(R0 As String) As String

  ExpR0 = ""
  ExpR0 = ExpR0 + Mid(R0, 32, 1)
  ExpR0 = ExpR0 + Mid(R0, 1, 1)
  ExpR0 = ExpR0 + Mid(R0, 2, 1)
  ExpR0 = ExpR0 + Mid(R0, 3, 1)
  ExpR0 = ExpR0 + Mid(R0, 4, 1)
  ExpR0 = ExpR0 + Mid(R0, 5, 1)
  ExpR0 = ExpR0 + Mid(R0, 4, 1)
  ExpR0 = ExpR0 + Mid(R0, 5, 1)
  ExpR0 = ExpR0 + Mid(R0, 6, 1)
  ExpR0 = ExpR0 + Mid(R0, 7, 1)
  ExpR0 = ExpR0 + Mid(R0, 8, 1)
  ExpR0 = ExpR0 + Mid(R0, 9, 1)
  ExpR0 = ExpR0 + Mid(R0, 8, 1)
  ExpR0 = ExpR0 + Mid(R0, 9, 1)
  ExpR0 = ExpR0 + Mid(R0, 10, 1)
  ExpR0 = ExpR0 + Mid(R0, 11, 1)
  ExpR0 = ExpR0 + Mid(R0, 12, 1)
  ExpR0 = ExpR0 + Mid(R0, 13, 1)
  ExpR0 = ExpR0 + Mid(R0, 12, 1)
  ExpR0 = ExpR0 + Mid(R0, 13, 1)
  ExpR0 = ExpR0 + Mid(R0, 14, 1)
  ExpR0 = ExpR0 + Mid(R0, 15, 1)
  ExpR0 = ExpR0 + Mid(R0, 16, 1)
  ExpR0 = ExpR0 + Mid(R0, 17, 1)
  ExpR0 = ExpR0 + Mid(R0, 16, 1)
  ExpR0 = ExpR0 + Mid(R0, 17, 1)
  ExpR0 = ExpR0 + Mid(R0, 18, 1)
```

```
  ExpR0 = ExpR0 + Mid(R0, 19, 1)
  ExpR0 = ExpR0 + Mid(R0, 20, 1)
  ExpR0 = ExpR0 + Mid(R0, 21, 1)
  ExpR0 = ExpR0 + Mid(R0, 20, 1)
  ExpR0 = ExpR0 + Mid(R0, 21, 1)
  ExpR0 = ExpR0 + Mid(R0, 22, 1)
  ExpR0 = ExpR0 + Mid(R0, 23, 1)
  ExpR0 = ExpR0 + Mid(R0, 24, 1)
  ExpR0 = ExpR0 + Mid(R0, 25, 1)
  ExpR0 = ExpR0 + Mid(R0, 24, 1)
  ExpR0 = ExpR0 + Mid(R0, 25, 1)
  ExpR0 = ExpR0 + Mid(R0, 26, 1)
  ExpR0 = ExpR0 + Mid(R0, 27, 1)
  ExpR0 = ExpR0 + Mid(R0, 28, 1)
  ExpR0 = ExpR0 + Mid(R0, 29, 1)
  ExpR0 = ExpR0 + Mid(R0, 28, 1)
  ExpR0 = ExpR0 + Mid(R0, 29, 1)
  ExpR0 = ExpR0 + Mid(R0, 30, 1)
  ExpR0 = ExpR0 + Mid(R0, 31, 1)
  ExpR0 = ExpR0 + Mid(R0, 32, 1)
  ExpR0 = ExpR0 + Mid(R0, 1, 1)

End Function

**********************************************************************

' Função que recebe uma string com 2 binários e retorna um decimal
equivalente
Function bin2ToDec(bin As String) As Byte

Dim pos1 As Byte
Dim pos6 As Byte

bin2ToDec = 0

  pos1 = CByte(Mid(bin, 1, 1))
  pos6 = CByte(Mid(bin, 2, 1))

  If pos1 = 1 Then
    bin2ToDec = bin2ToDec + 2
  End If

  If pos6 = 1 Then
    bin2ToDec = bin2ToDec + 1
  End If

End Function
```

```
**********************************************************************

' Função que recebe uma string com 4 binários e retorna um decimal
equivalente
Function bin4ToDec(bin As String) As Byte

Dim pos2 As Byte
Dim pos3 As Byte
Dim pos4 As Byte
Dim pos5 As Byte

bin4ToDec = 0

  pos2 = CByte(Mid(bin, 1, 1))
  pos3 = CByte(Mid(bin, 2, 1))
  pos4 = CByte(Mid(bin, 3, 1))
  pos5 = CByte(Mid(bin, 4, 1))

  If pos2 = 1 Then
    bin4ToDec = bin4ToDec + 8
  End If

  If pos3 = 1 Then
    bin4ToDec = bin4ToDec + 4
  End If

   If pos4 = 1 Then
    bin4ToDec = bin4ToDec + 2
  End If

  If pos5 = 1 Then
    bin4ToDec = bin4ToDec + 1
  End If

End Function

**********************************************************************

Function dec2Bin(DecimalIn As Byte, NumberOfBits As Variant) As String

   dec2Bin = ""
   DecimalIn = Int(CDec(DecimalIn))
   Do While DecimalIn <> 0
      dec2Bin = Format$(DecimalIn - 2 * Int(DecimalIn / 2)) & dec2Bin
      DecimalIn = Int(DecimalIn / 2)
   Loop
   If Not IsMissing(NumberOfBits) Then
```

```
      If Len(dec2Bin) > NumberOfBits Then
        dec2Bin = "Error - Number exceeds specified bit size"
      Else
        dec2Bin = Right$(String$(NumberOfBits, "0") & dec2Bin, NumberOfBits)
      End If
    End If

End Function

**********************************************************************

Function FS1(b As String) As String
  Dim varB1 As String
  Dim varB2 As String
  Dim decB1 As Byte
  Dim decB2 As Byte
  Dim resultadoDecimal As Byte

  'calcula a linha da matriz S1 pegando o primeiro e o ultimo caracter da
cadeia de 6 caracteres B
  varB1 = Mid(b, 1, 1) + Mid(b, 6, 1)
  'Tranforma uma string de 2 dígitos binários calculados anteriormente em um
número decimal
  decB1 = bin2ToDec(varB1)

  'calcula a coluna da matriz S1
  varB2 = Mid(b, 2, 1) + Mid(b, 3, 1) + Mid(b, 4, 1) + Mid(b, 5, 1)
  decB2 = bin4ToDec(varB2)

  'Recupera resultado da matriz S
  resultadoDecimal = MatrizS1(decB1, decB2)

  'Transforma esse decimal em um binário de 4 bits
  'FS1 = decToBin(resultadoDecimal, 4)
   FS1 = dec2Bin(resultadoDecimal, 4)

End Function

Function FS2(b As String) As String
  Dim varB1 As String
  Dim varB2 As String
  Dim decB1 As Byte
  Dim decB2 As Byte
  Dim resultadoDecimal As Byte

  'calcula a linha da matriz S1
  varB1 = Mid(b, 1, 1) + Mid(b, 6, 1)
```

```
    decB1 = bin2ToDec(varB1)

    'calcula a coluna da matriz S1
    varB2 = Mid(b, 2, 1) + Mid(b, 3, 1) + Mid(b, 4, 1) + Mid(b, 5, 1)
    decB2 = bin4ToDec(varB2)

    'Recupera resultado da matriz S
    resultadoDecimal = MatrizS2(decB1, decB2)

    'Transforma esse decimal em um binário de 4 bits
    'FS1 = decToBin(resultadoDecimal, 4)
     FS2 = dec2Bin(resultadoDecimal, 4)

End Function

Function FS3(b As String) As String
    Dim varB1 As String
    Dim varB2 As String
    Dim decB1 As Byte
    Dim decB2 As Byte
    Dim resultadoDecimal As Byte

    'calcula a linha da matriz S1
    varB1 = Mid(b, 1, 1) + Mid(b, 6, 1)
    decB1 = bin2ToDec(varB1)

    'calcula a coluna da matriz S1
    varB2 = Mid(b, 2, 1) + Mid(b, 3, 1) + Mid(b, 4, 1) + Mid(b, 5, 1)
    decB2 = bin4ToDec(varB2)

    'Recupera resultado da matriz S
    resultadoDecimal = MatrizS3(decB1, decB2)

    'Transforma esse decimal em um binário de 4 bits
    'FS1 = decToBin(resultadoDecimal, 4)
     FS3 = dec2Bin(resultadoDecimal, 4)

End Function

Function FS4(b As String) As String
   Dim varB1 As String
    Dim varB2 As String
    Dim decB1 As Byte
    Dim decB2 As Byte
    Dim resultadoDecimal As Byte

    'calcula a linha da matriz S1
```

```
  varB1 = Mid(b, 1, 1) + Mid(b, 6, 1)
  decB1 = bin2ToDec(varB1)

  'calcula a coluna da matriz S1
  varB2 = Mid(b, 2, 1) + Mid(b, 3, 1) + Mid(b, 4, 1) + Mid(b, 5, 1)
  decB2 = bin4ToDec(varB2)

  'Recupera resultado da matriz S
  resultadoDecimal = MatrizS4(decB1, decB2)

  'Transforma esse decimal em um binário de 4 bits
  'FS1 = decToBin(resultadoDecimal, 4)
   FS4 = dec2Bin(resultadoDecimal, 4)

End Function

Function FS5(b As String) As String
  Dim varB1 As String
  Dim varB2 As String
  Dim decB1 As Byte
  Dim decB2 As Byte
  Dim resultadoDecimal As Byte

  'calcula a linha da matriz S1
  varB1 = Mid(b, 1, 1) + Mid(b, 6, 1)
  decB1 = bin2ToDec(varB1)

  'calcula a coluna da matriz S1
  varB2 = Mid(b, 2, 1) + Mid(b, 3, 1) + Mid(b, 4, 1) + Mid(b, 5, 1)
  decB2 = bin4ToDec(varB2)

  'Recupera resultado da matriz S
  resultadoDecimal = MatrizS5(decB1, decB2)

  'Transforma esse decimal em um binário de 4 bits
  'FS1 = decToBin(resultadoDecimal, 4)
   FS5 = dec2Bin(resultadoDecimal, 4)

End Function

Function FS6(b As String) As String
  Dim varB1 As String
  Dim varB2 As String
  Dim decB1 As Byte
  Dim decB2 As Byte
 Dim resultadoDecimal As Byte
```

```
  'calcula a linha da matriz S1
  varB1 = Mid(b, 1, 1) + Mid(b, 6, 1)
  decB1 = bin2ToDec(varB1)

  'calcula a coluna da matriz S1
  varB2 = Mid(b, 2, 1) + Mid(b, 3, 1) + Mid(b, 4, 1) + Mid(b, 5, 1)
  decB2 = bin4ToDec(varB2)

  'Recupera resultado da matriz S
  resultadoDecimal = MatrizS6(decB1, decB2)

  'Transforma esse decimal em um binário de 4 bits
  'FS1 = decToBin(resultadoDecimal, 4)
   FS6 = dec2Bin(resultadoDecimal, 4)

End Function

Function FS7(b As String) As String
  Dim varB1 As String
  Dim varB2 As String
  Dim decB1 As Byte
  Dim decB2 As Byte
  Dim resultadoDecimal As Byte

  'calcula a linha da matriz S1
  varB1 = Mid(b, 1, 1) + Mid(b, 6, 1)
  decB1 = bin2ToDec(varB1)

  'calcula a coluna da matriz S1
  varB2 = Mid(b, 2, 1) + Mid(b, 3, 1) + Mid(b, 4, 1) + Mid(b, 5, 1)
  decB2 = bin4ToDec(varB2)

  'Recupera resultado da matriz S
  resultadoDecimal = MatrizS7(decB1, decB2)

  'Transforma esse decimal em um binário de 4 bits
  'FS1 = decToBin(resultadoDecimal, 4)
   FS7 = dec2Bin(resultadoDecimal, 4)

End Function

Function FS8(b As String) As String
  Dim varB1 As String
  Dim varB2 As String
  Dim decB1 As Byte
  Dim decB2 As Byte
  Dim resultadoDecimal As Byte
```

```
    'calcula a linha da matriz S1
    varB1 = Mid(b, 1, 1) + Mid(b, 6, 1)
    decB1 = bin2ToDec(varB1)

    'calcula a coluna da matriz S1
    varB2 = Mid(b, 2, 1) + Mid(b, 3, 1) + Mid(b, 4, 1) + Mid(b, 5, 1)
    decB2 = bin4ToDec(varB2)

    'Recupera resultado da matriz S
    resultadoDecimal = MatrizS8(decB1, decB2)

    'Transforma esse decimal em um binário de 4 bits
    'FS1 = decToBin(resultadoDecimal, 4)
     FS8 = dec2Bin(resultadoDecimal, 4)

End Function
```

Obs.: O algoritmo DES deste estudo foi implementado na linguagem VBA existente dentro do programa Ms-Excel.

APÊNDICE B – Implementação da aplicação MotBank

I – Tela da aplicação MotBank

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
android:paddingBottom="@dimen/activity_vertical_margin"
android:orientation="vertical"
tools:context=".MainActivity">

<LinearLayout
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="1"
android:orientation="horizontal">

<ImageView
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="1"
android:src="@drawable/mot"
android:scaleType="fitXY"
/>

<LinearLayout
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="2"
android:orientation="vertical">

<ImageView
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="2"
android:scaleType="fitXY"
android:src="@drawable/bank"/>
<TextView
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="1"
android:textSize="15sp"
android:typeface="serif"
android:textStyle="bold"
android:textColor="#DAC22A"
android:text="@string/register"/>
```

```
</LinearLayout>

</LinearLayout>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="1"
android:orientation="horizontal"
>

<TextView
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="1"
android:gravity="center"
android:text="AGÊNCIA"
android:textSize="20sp"
android:typeface="serif"
android:textStyle="bold"
/>

<EditText
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="1"
android:gravity="center"
android:background="@android:drawable/editbox_background_normal
"
/>

</LinearLayout>

<View
android:layout_width="match_parent"
android:layout_height="1dp"
android:layout_weight="0.01"
android:background="@android:color/darker_gray"/>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="1"
android:orientation="horizontal">

<TextView
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="1"
```

```
android:gravity="center"
android:text="CONTA"
android:textSize="20sp"
android:typeface="serif"
android:textStyle="bold"
/>

<EditText
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="1"
android:gravity="center"
android:background="@android:drawable/editbox_background_normal
"

/>

</LinearLayout>

<View
android:layout_width="match_parent"
android:layout_height="1dp"
android:layout_weight="0.01"
android:background="@android:color/darker_gray"/>


<LinearLayout
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="1"
android:orientation="horizontal"
android:id="@+id/laySenha"
>

<TextView
android:id="@+id/txtSenha"
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="1"
android:gravity="center"
android:text="SENHA"
android:textSize="20sp"
android:typeface="serif"
android:textStyle="bold"
/>

<EditText
android:id="@+id/edtSenha"
android:layout_width="0dp"
android:layout_height="match_parent"
```

```
android:layout_weight="1"
android:gravity="center"
android:password="true"
android:background="@android:drawable/editbox_background_normal
"
/>


</LinearLayout>


<RelativeLayout
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="0.5"
android:orientation="horizontal"
android:clickable="true"
android:background="@drawable/corners"
android:id="@+id/layBotao"
>


<Button
android:id="@+id/button1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:drawableLeft="@drawable/cifrao"
android:background="@drawable/corners"
android:layout_marginLeft="120dp"
android:drawablePadding="10dp"
android:layout_centerInParent="true"
android:text="ACESSAR CONTA">
        Button>

</Button>
</RelativeLayout>

</LinearLayout>
```

II – Tela da aplicação rodando no servidor da instituição

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
android:paddingBottom="@dimen/activity_vertical_margin"
android:orientation="vertical"
```

```
tools:context=".MainActivity">

<LinearLayout
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="1"
android:orientation= "horizontal">

<TextView
android:layout_width="match_parent"
android:layout_height="match_parent"
android:gravity="center"
android:textSize="12sp"
android:text="Informações para escolha dos números primos"
/>

</LinearLayout>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="1"
android:orientation= "horizontal">

<TextView
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="1"
android:gravity="center"
android:textSize="15sp"
android:text="Nome da Mãe"
/>
<EditText
android:id="@+id/txtInfo1"
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="1"
android:gravity="center"
android:textSize="18sp"
android:background="@android:drawable/editbox_background_normal"
/>

</LinearLayout>


<LinearLayout
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="1"
android:orientation= "horizontal">
```

```
<TextView
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="1"
android:gravity="center"
android:textSize="15sp"
android:text="Data de Nascimento"
/>
<EditText
android:id="@+id/txtInfo2"
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="1"
android:gravity="center"
android:textSize="18sp"
android:background="@android:drawable/editbox_background_normal"
/>

</LinearLayout>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="1"
android:orientation= "horizontal">

<TextView
android:id="@+id/lblPrimo1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:gravity="center"
android:textSize="15sp"
android:text="Primo 1"
/>

<EditText
android:id="@+id/txtPrimo1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:gravity="center"
android:layout_below="@id/lblPrimo1"
android:textSize="15sp"
android:background="@android:drawable/editbox_background_normal"
/>

<TextView
android:id="@+id/lblPrimo2"
```

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:gravity="center"
android:layout_toRightOf="@id/lblPrimo1"
android:textSize="15sp"
android:text="Primo 1"
/>


<EditText
android:id="@+id/txtPrimo2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:gravity="center"
android:layout_below="@id/lblPrimo2"
android:textSize="15sp"
android:background="@android:drawable/editbox_background_normal"
/>


</LinearLayout>



<LinearLayout
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="1.5"
android:orientation= "horizontal">

<Button
android:id="@+id/btnGerar"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginLeft="100dp"
android:layout_centerInParent="true"
android:text="Gerar"
android:singleLine="true">

</Button>

</LinearLayout>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="1"
```

```
android:orientation= "horizontal">

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:gravity="center"
android:layout_toRightOf="@id/lblPrimo1"
android:textSize="20sp"
android:text="n"
/>

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:gravity="center"
android:layout_toRightOf="@id/lblPrimo1"
android:textSize="20sp"
android:text="e"
/>

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:gravity="center"
android:layout_toRightOf="@id/lblPrimo1"
android:textSize="15sp"
android:text="d"
/>

</LinearLayout>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="2"
android:orientation= "horizontal">

<EditText
android:id="@+id/txtN"
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="1"
android:gravity="center"
android:textSize="20sp"
android:background="@android:drawable/editbox_background_normal"
/>
```

```
<EditText
android:id="@+id/txtE"
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="1"
android:gravity="center"
android:textSize="20sp"
android:background="@android:drawable/editbox_background_normal"
/>

<EditText
android:id="@+id/txtD"
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="1"
android:gravity="center"
android:textSize="20sp"
android:background="@android:drawable/editbox_background_normal"
/>



</LinearLayout>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="1"
android:orientation= "horizontal">

<TextView
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="1"
android:gravity="center"
android:textSize="15sp"
android:text="Senha Clara"
/>
<EditText
android:id="@+id/txtSenha"
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="1"
android:gravity="center"
android:textSize="15sp"
android:background="@android:drawable/editbox_background_normal"
/>

</LinearLayout>
```

```
<LinearLayout
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="1.5"
android:orientation= "horizontal">

<TextView
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="1"
android:gravity="center"
android:textSize="15sp"
android:text="Senha ASCii"
/>
<EditText
android:id="@+id/txtSenhaAscii"
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="1"
android:gravity="center"
android:textSize="15sp"
android:background="@android:drawable/editbox_background_normal"
/>

</LinearLayout>


<LinearLayout
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="1.5"
android:orientation= "horizontal">

<TextView
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="1"
android:gravity="center"
android:textSize="15sp"
android:text="Senha Criptografada"
/>
<EditText
android:id="@+id/txtSenhaCripto"
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="1"
android:gravity="center"
android:textSize="15sp"
android:background="@android:drawable/editbox_background_normal"
/>
```

```
</LinearLayout>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="2"
android:orientation= "horizontal">

<TextView
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="1"
android:gravity="center"
android:textSize="15sp"
android:text="Senha Novamente"
/>
<EditText
android:id="@+id/txtSenhaNova"
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="1"
android:gravity="center"
android:textSize="15sp"
android:background="@android:drawable/editbox_background_normal"
/>

</LinearLayout>


<LinearLayout
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="1.5"
android:orientation= "horizontal">

<Button
android:id="@+id/btnCodificar"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerInParent="true"
android:text="Codificar"
android:singleLine="true">

</Button>

<Button
android:id="@+id/btnDecodificar"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
```

```
android:layout_marginLeft="50dp"
android:layout_centerInParent="true"
android:text="Decodificar"
android:singleLine="true">
</Button>

</LinearLayout>


</LinearLayout>
```

III – Código:

```
package br.com.motbank.rsa;

import android.content.ActivityNotFoundException;
import android.content.Intent;
import android.provider.MediaStore;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import java.math.BigInteger;
import java.util.ArrayList;

public class MainActivity extends AppCompatActivity implements
View.OnClickListener {

private Integer tamCadeia;
private EditText n;
private EditText d;
private EditText e;
private EditText senhaNova;
private EditText decodifica;
private EditText codifica;
private EditText senhaASCii;
private EditText txtPrimo1;
private EditText txtPrimo2;
private Double qq;
private Button btnGerar;
private Button btnCodificar;
private Button btnDecodificar;
private ArrayList<Integer>divisores;
private ArrayList<Integer>binarios;
```

```
private ArrayList<String>cadeia;
private ArrayList<String>cadeiaCifrada;
private ArrayList<String>cadeiaDecifrada;
private Integer ascMae;
private Integer ascDtNasc;
private Integer primo1;
private Integer primo2;
private EditText txtInfo1;
private EditText txtInfo2;
//controla se foram acrescentados zeros à direita da última cadeia.
private Integer ultimo;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
n = (EditText)findViewById(R.id.txtN);
d = (EditText)findViewById(R.id.txtD);
e = (EditText)findViewById(R.id.txtE);
txtInfo1 = (EditText)findViewById(R.id.txtInfo1);
txtInfo2 = (EditText)findViewById(R.id.txtInfo2);
ultimo=0;

txtPrimo1 = (EditText)findViewById(R.id.txtPrimo1);
txtPrimo2 = (EditText)findViewById(R.id.txtPrimo2);
decodifica = (EditText)findViewById(R.id.txtSenhaCripto);
codifica = (EditText)findViewById(R.id.txtSenha);
senhaNova = (EditText)findViewById(R.id.txtSenhaNova);
senhaASCii = (EditText)findViewById(R.id.txtSenhaAscii);
divisores = new ArrayList<Integer>();
btnGerar = (Button)findViewById(R.id.btnGerar);
btnGerar.setOnClickListener(this);

btnCodificar = (Button)findViewById(R.id.btnCodificar);
btnCodificar.setOnClickListener(this);

btnDecodificar = (Button)findViewById(R.id.btnDecodificar);
btnDecodificar.setOnClickListener(this);

    }

//*********************************************************************

//Esta função soma os valores ASCII de cada caracter do parâmetro. No final gera
um numero inteiro que servirá de base para
 // encontrar um primo
private Integer somaAsc(String cadastro){
        Integer tempAsc = 0;
for (int i=0;i<cadastro.length();i++){
            tempAsc = tempAsc + (int) cadastro.charAt(i);
```

```
        }
return tempAsc;
    }

//**********************************************************************


//Esta função converte seu parâmetro em uma sequencia ASC de seus caracteres

private String converteAsc(String original){
        String tempAsc = "";
for (int i=0;i<original.length();i++){
int asc = original.charAt(i);
            tempAsc+=String.valueOf(asc);
        }
//verifica se precisará acrescentar zero ao final da string
if ((tempAsc.length() % tamCadeia) != 0 ){
ultimo = tamCadeia - (tempAsc.length() % tamCadeia);
for (int j=0;j<ultimo;j++){
                tempAsc+="0";
            }
        }


return tempAsc;
    }

//**********************************************************************


//divide a cadeia ascii em subcadeias de mesmo tamanho
private ArrayList <String> geraCadeia(String mensagemASC) {
        ArrayList<String> temp = new ArrayList<String>();
int i;
//calcula em quantos blocos a mensagem convertida em ascii será dividida
Integer tam = mensagemASC.length() / tamCadeia;
if ( (mensagemASC.length() % tamCadeia) !=0 )
            tam++;

for (i=0;i<tam;i++) {
            Integer inicioCadeia = tamCadeia*i;
            Integer finalCadeia = inicioCadeia + (tamCadeia-1);
            temp.add(i,mensagemASC.substring(inicioCadeia, finalCadeia+1));
        }

return temp;
    }


//**********************************************************************
```

```
//transforma a cadeia original em cadeia cifrada
private ArrayList <String> geraCadeiaCifrada(ArrayList<String> cadeiaAntes) {
	ArrayList<String> temp = new ArrayList<String>();
	String resultado;


for (int i=0;i<cadeiaAntes.size();i++) {
		resultado =  String.valueOf((int)potencia(Integer.parseInt(cadeiaAntes.get(i)),
Integer.parseInt(e.getText().toString()), Integer.parseInt(n.getText().toString())));
// verifica se o resultado da expressão é um resto menor que o tamanho padrão de
cada cadeia.
if (resultado.length() <tamCadeia) {
			StringBuilder resultadoAcrescenta = new
StringBuilder(String.valueOf(resultado));
			Integer acrescenta = tamCadeia - resultado.length();
for (int j=1;j<=acrescenta;j++){
			  resultadoAcrescenta.insert(0,"0");
			}
			resultado = resultadoAcrescenta.toString();
		}
			temp.add(i, resultado);
	}

return temp;
   }

//**********************************************************************

// transforma a cadeia codificada na cadeia original
private ArrayList <String> geraCadeiaDecifrada(ArrayList<String> cadeiaAntes) {
	ArrayList<String> temp = new ArrayList<String>();
	String resultado;


for (int i=0;i<cadeiaAntes.size();i++) {
		resultado =  String.valueOf((int)potencia(Integer.parseInt(cadeiaAntes.get(i)),
Integer.parseInt(d.getText().toString()), Integer.parseInt(n.getText().toString())));
// verifica se o resultado da expressão é um resto menor que o tamanho padrão de
cada cadeia.
if (resultado.length() <tamCadeia) {
			StringBuilder resultadoAcrescenta = new
StringBuilder(String.valueOf(resultado));
			Integer acrescenta = tamCadeia - resultado.length();
for (int j=1;j<=acrescenta;j++){
			  resultadoAcrescenta.insert(0,"0");
			}
			resultado = resultadoAcrescenta.toString();
		}
		temp.add(i, resultado);
```

```
        }

return temp;
    }


//**********************************************************************

//calcula o módulo
public double potencia(double a, Integer b, Integer n){
double r = (double)1;
        Boolean controle = true;
while (controle) {

if (b % 2 == 1)
                r = r * a % n;

            b /= 2;

if (b == 0)
break;

            a = a * a % n;
        }


return r;
    }

public void onClick(View v) {
//gera os números primos
if (v.getId() == R.id.btnGerar) {

ascMae = somaAsc(txtInfo1.getText().toString());
ascDtNasc = somaAsc(txtInfo2.getText().toString());
primo1 = encontraPrimo(ascMae);
primo2 = encontraPrimo(ascDtNasc);
            Integer p = primo1;
            Integer q = primo2;
txtPrimo1.setText(String.valueOf(p));
txtPrimo2.setText(String.valueOf(q));
n.setText(String.valueOf(p*q));
tamCadeia = n.getText().toString().length() - 1;
qq = (double)((p-1) * (q-1));
divisores = Fatoracao(qq);
double chaveE = funcaoE(qq);
e.setText(String.valueOf((int)chaveE));
double chaveD = Euclides_ext(chaveE, qq, 1);
d.setText(String.valueOf((int)chaveD));
        }
```

```
//codifica a senha

if (v.getId() == R.id.btnCodificar) {
        Double resultado = (double)0;
        Double potencia;
        String cifrada = "";
cadeia = new ArrayList<String>();
String mensagemASC = converteAsc (codifica.getText().toString());

cadeia = geraCadeia (mensagemASC);

cadeiaCifrada = geraCadeiaCifrada(cadeia);

for (int i = 0; i<cadeiaCifrada.size();i++){
            cifrada = cifrada + String.valueOf(cadeiaCifrada.get(i));
        }


decodifica.setText(cifrada);
        cifrada="";
for (int i = 0; i<mensagemASC.length();i++){
            cifrada = cifrada + mensagemASC.charAt(i);
        }

senhaASCii.setText(cifrada);
    }


//**********************************************************************

//decodifica a senha
if (v.getId() == R.id.btnDecodificar) {
        Double resultado = (double)0;
        Double potencia;
        String decifrada = "";
cadeia = new ArrayList<String>();

cadeiaDecifrada = geraCadeiaDecifrada (cadeiaCifrada);


for (int i = 0; i<cadeiaDecifrada.size();i++){
            decifrada = decifrada + String.valueOf(cadeiaDecifrada.get(i));
        }


senhaNova.setText(decifrada);
    }
  }
```

```
//*********************************************************************


   //Busca um primo a partir do resultado ASCII do cadastro variando para baixo. Assim pega sempre o maior primo possível.
public Integer encontraPrimo(Integer cadastro){
      Integer resultado = 1;

for (int i = cadastro; i>2; i--){
if (verificaDivisores(i) <3){
            resultado = i;
break;
         }
      }

return resultado;
   }

//*********************************************************************


//calcula todos os divisores de um número
public Integer verificaDivisores (Integer numero){
//Todo numero tem pelo menos 2 divisores: 1 e ele mesmo.
Integer divisores=2;

for (int j=2;j<numero;j++){
if (numero%j == 0){
            divisores++;
break;
         }
      }

return divisores;
   }


//*********************************************************************


// calcula a exponenciação dada uma base e um expoente
public Double Expoente (Integer base, Integer exp){
if (exp == 0 ) {
return (double)1;
   }
else {
return base * Expoente(base, exp - 1);
   }
}
```

```
//*********************************************************************


//algoritmo de Euclides estendido
public Double Euclides_est (Double a, Double b, Integer c){
        Double r;

        r = resto(b, a);

if (r == 0) {
return resto((c / a), (b / a));
        }
else {
return ((Euclides_ext(r, a, -c) * b + c) / resto(a, b));
        }

    }

//*********************************************************************


//função utilizada no algoritmo de Euclides estendido
public Double resto(Double a, Double b)
    {
        Double r = a % b;

/* Uma correção é necessária se r e b não forem do mesmo sinal */

    /* se r for negativo e b positivo, precisa corrigir */
if ((r <0) && (b >0))
return (b + r);

/* Se ra for positivo e b negativo, nova correcao */
if ((r >0) && (b <0))
return (b + r);

return (r);
    }

//*********************************************************************


//função que calcula a chave E
public Double funcaoE (Double qq){
        Double retorno = 0.0;
        Boolean checa = false;

//  numero de divisores de qq
Integer ind = divisores.size();
```

```
for (int i=2;i<=qq;i++){

//Percorre os elementos do vetor de divisores de qq
for (int j=0;j<ind;j++) {

// Se for divisível por qualquer elemento da fatoração entao este número já não é primo entre si com qq
if ((i % divisores.get(j))== 0)
break;

//Último elemento da lista. Se ele não for divisível por 0 então o número i é primo entre si com qq
if (j == (ind - 1))
if ((i % divisores.get(j)) != 0){
                    checa = true;
                    retorno = (double) i;
                }
        }

//Se já foi achado um divisor sai do loop
if (retorno != 0)
break;
      }
return retorno;
  }


//*********************************************************************
//Este método encontrará os divisores de qq
public ArrayList<Integer> Fatoracao (Double qq){
       ArrayList<Integer> temp = new ArrayList<Integer>();
//tamanho inicial do vetor de divisores

       //este for procurará os divisores começando por 2
for (int i=2;i<=40;i++){
//Se o resto da divisão for zero então é um divisor de qq e adiciona no vetor de divisores
while ((qq % i) == 0){
                temp.add(i);
                qq = qq / i;
            }
// Todos os divisores já foram achados então pode-se sair da busca
if (qq < i)
break;
        }

return temp;
   }
}
```